UNIVERSITY OF OSLO
Department of Informatics

The SCORE method and tool

A tool-supported method for risk estimation using CORAS diagrams

Master thesis

Stig Torsbakken

May 2nd 2007

Abstract

Security incidents targeting all kinds of computerised systems occur on a daily basis within most organisations and vulnerability records increase steadily. The modern industry hence demands secure and reliable systems and calls for good methods for security risk analysis in order to identify the threat picture. CORAS is a method for conducting security risk analysis, and similar to most security risk analysis methods, it may involve participants from totally different professions and background that hardly speak the same language.

We have developed a tool-supported method, SCORE, that aims to deal with problems related to one of the sub-processes of a CORAS security risk analysis, namely risk estimation. SCORE applies to CORAS diagrams, a special purpose graphical language designed to facilitate communication between roles of different professions, that are used to describe risk and threat scenarios.

This thesis presents and documents the results from the SCORE project. Its main contribution is the SCORE tool-supported method that consists of two distinct, but closely related modules; the SCORE method and the SCORE tool. The first defines rules for processing input from the risk estimation process in a structured fashion and standardises the risk estimation process of a CORAS security risk analysis. The latter is a computerised tool integrated with the existing CORAS tool designed to support the SCORE method by implementing its defined rules.

The SCORE tool-supported method has successfully been evaluated on a CORAS security risk analysis supported by SCORE. Together with formal evaluation techniques, SCORE has come to the conclusion that a carefully designed tool-supported method will increase the efficiency of risk estimation using CORAS diagrams.

The thesis furthermore provides documentation of how to execute the SCORE method, and documentation in terms of software requirements and architectural design such that the SCORE tool may be fully extended to a commercial implementation into the CORAS tool.

Acknowledgements

I would like to express my gratitude to first of all my excellent tutor, professor Ketil Stølen, for his genuine interest in my work and always presence throughout the whole of this project. His professionalism, guidance and contribution has assured for the quality of my work. Even more importantly, his great expectations to me and ever challenging nature has pushed me, inspired me and always acquired the best of me. Thank you.

My thanks go also to SINTEF for allowing me to take part in one of their industrial CORAS security risk analyses that has proven to be crucial to my understanding of CORAS and my problem description. Especially thanks to Ketil Stølen again, now as my boss, Iselin Engan and Mass Soldal Lund for involving me and trusting me with responsibility for the security risk analysis. Furthermore, thanks to SINTEF for letting me take part in the development of the CORAS tool, and Fredrik Vraalsen for expertise guidance on the CORAS tool.

I wish to thank Jan Magne Stubrud for his cooperation and that he willingly let me perform a security risk analysis for his business Stubrud sjakk-shop.

Last but not least, thanks to the people I surround myself with on a daily basis. My fellow students at the institute for informatics; Rudin, Pål, Kjetil, Nikolaj, Håkon, Christer, Olav and the whole of ProsIT; our numerous discussions – inspiring as well as frustrating – it would not been the same without. To my house mates, my family and my ever supporting girlfriend, Hege; thanks for putting up with me day and night.

Oslo, May 2007

Stig Torsbakken

Contents

Contents **vii**

1 Introduction **1**

1.1 Motivation 2

1.2 Contribution 2

1.3 Thesis structure 4

2 Background **7**

2.1 Security risk analysis 8

2.2 Established threat and risk modelling techniques 9

2.2.1 Fault tree analysis (FTA) 9

2.2.2 Event tree analysis (ETA) 13

2.2.3 Attack tree analysis (ATA) 15

2.3 CORAS diagrams 16

2.3.1 Syntax 17

2.3.2 Pragmatics 20

2.3.3 Semantics 21

3 Problem analysis **25**

3.1 Aim of the problem analysis 25
3.2 Case study 26
3.3 Problem characterisation 27
3.3.1 Overall problem description 27
3.3.2 Motivation 28
3.3.3 Refined problem description 28
3.4 Discussion 30
3.4.1 Established modelling techniques and CORAS diagrams . 30
3.4.2 Developing the SCORE method 33
3.4.3 Developing the SCORE tool 34
3.5 Research strategy 35
3.5.1 Software engineering and technology research 35
3.5.2 Research strategy techniques 36
3.6 Detailed success criteria 37
3.6.1 Success criteria for the SCORE method 37
3.6.2 Success criteria for the SCORE tool 37
3.6.3 Success criteria for SCORE in CORAS 38

4 The SCORE method **39**
4.1 Legend 40
4.2 Quantitative rules 41
4.2.1 1-to-1 complete relationships 41
4.2.2 Many-to-1 complete relationships 42
4.2.3 Incomplete relationships 43
4.2.4 1-to-1 complete paths 44

4.3 Qualitative rules 45
4.3.1 Defining a set 45
4.3.2 Set multiplication 46
4.3.3 Set addition 47
4.3.4 Selecting the correct interval 47
4.3.5 Inconsistencies 47

5 Requirements for the tool **49**
5.1 The vision 49
5.1.1 Purpose 50
5.2 Software requirements 51
5.2.1 User requirements 51
5.2.2 System requirements 53

6 The SCORE tool **55**
6.1 How the tool implements the SCORE method 56
6.2 Graphical user interface (GUI) 56
6.3 Design 61
6.3.1 The SCORE tool and the CORAS tool 62
6.3.2 The SCORE tool architectural design 63
6.3.3 The SCORE tool behavioural design 65
6.4 Implementation 70

7 Evaluation **73**
7.1 Case study 73
7.1.1 Stubrud sjakk shop security risk analysis 74

7.1.2 Case study experiences 84
7.2 Evaluation of the SCORE method 86
7.3 The SCORE tool testing 92

8 Fulfilment of success criteria **95**
8.1 Detailed success criteria 96
8.1.1 The SCORE method 96
8.1.2 The SCORE tool 97
8.1.3 The SCORE tool-supported method in CORAS 99
8.1.4 Summary 99
8.2 Overall success criteria 99
8.3 Overall hypothesis 101

9 Discussion **103**
9.1 Assumptions and considerations 103
9.1.1 Selection of background material 104
9.1.2 Chosen research techniques 104
9.1.3 Designated SCORE method rules 105
9.1.4 Design of the SCORE tool 105
9.1.5 Application of SCORE in the evaluation 105
9.2 Threats to the validity of results 106
9.2.1 Quality assurance of SCORE requirements 106
9.2.2 Liability of field trial 106
9.2.3 Research strategies – empirical investigations 107
9.2.4 Liability of experimental simulation 107
9.3 Validation 108

10 Conclusion **109**

10.1 Findings 110

10.2 Future work 110

10.2.1 The SCORE method 110

10.2.2 The SCORE tool 111

Bibliography **113**

Appendices **117**

A The SCORE tool architectural design **119**

B Software requirements specification **135**

C Security risk analysis of Stubrud sjakk-shop **147**

D The SCORE tool test results **179**

List of Symbols and Abbreviations **185**

List of Figures **186**

List of Tables **189**

Chapter 1

Introduction

Computer security is undoubtedly one of the fast-growing areas in today's rapidly changing IT industry. As technology gets more and more complex and providing us with endless opportunities and features, it opens doors also to those who come with malignant motives. In 2006 IBM Internet Security Systems X-Force®research and development team registered a 39.5 percent increase over 2005 in new vulnerability records and expects the trend to continue also in 2007 [1]. In order to face the rising demands of modern times, the industry calls for means for security risk management to protect their business. A security risk analysis may then be the foremost instrument to identify the threat picture in order to decide what security technologies to implement.

While security risk management is an increasing priority in many organisations today, we see that generally in IT projects and no less for those related to security risk analysis, IT professionals and their clients hardly speak the same language [2]. A security risk analysis involves experts from completely different professions and backgrounds. The security risk analysis targets systems that may be of great complexity, thus requiring exhaustive cooperation between the participants involved. This brings about the demand for well-developed methods for conducting security risk analyses and for supporting tools to make the execution of the analysis perform painlessly.

CORAS [3] is a method for conducting security risk analysis. Designed with the purpose to facilitate communication between roles of different professions, CORAS offers a special purpose graphical language (so-called CORAS diagrams) to describe what we call risk and threat scenarios. A CORAS security risk analysis consists of five phases, each of which makes use of structured brainstorming to

extract information about the system under evaluation needed to create the CORAS diagrams. The result of a risk analysis is an identified threat picture with evaluated risks of which some may require treatment.

This work, from now referred to as *the SCORE project*, focuses on one of the sub-processes of a CORAS security risk analysis, namely the process of risk estimation. This is the phase where the identified risks are evaluated with respect to likelihood and consequence values. When accomplished, the results of this phase allow the risks to be evaluated with respect to whether they should be treated or be accepted. The SCORE project aims to overcome the problem of practical and theoretical performance in the estimation of likelihood values.

The outcome of the SCORE project will be a tool-supported method. With the SCORE integrated CORAS, we believe the risk analysis participants will find it easier to pin their expertise towards the core of the risk estimation problems and not be distracted or confused by practicality and comprehensibility obstacles.

1.1 Motivation

In order for the risk analysts involved to lead and execute an optimum security risk analysis, they should have the best tools and methods adapted to accommodate their needs. CORAS does not fulfil these needs when it comes to the risk estimation phase. Studies show that during risk estimation in CORAS, there is often confusion and too much conjecture that may lead to inaccuracies in estimated values [4]. There is no tool-support for combining or processing the input from the field experts and no clearly defined method for calculating probabilities. Furthermore, CORAS does not properly account for how to treat qualitative input combined with precise, numerical, quantitative data.

The main objective for the SCORE project is to develop a special purpose method and tool to support the risk estimation phase of CORAS.

1.2 Contribution

The SCORE project has developed a tool-supported method to be used in the risk estimation phase of a CORAS security risk analysis. The tool-supported method consists or two distinct modules; *the SCORE method* and *the SCORE tool*, where the SCORE tool relies and uses the results from the SCORE method (illustrated in figure 1.1.

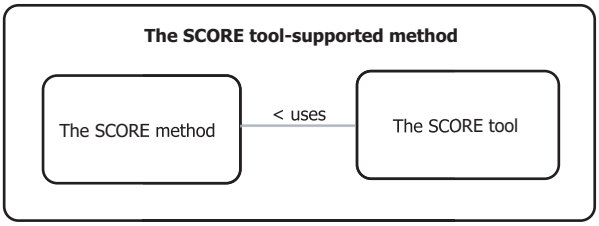

Figure 1.1: The SCORE project overview

The SCORE method defines how to perform the risk estimation based on CORAS diagrams. The method is formalised as a set of rules. Created with the intention of being supported by the tool, the rules assemble the basis upon which the tool is implemented. The method nevertheless constitutes a unique and stand-alone contribution of this project, evaluated as such. In theory, the method could be applied to a risk analysis manually with the rule definitions in hand. However, such employment of the method could be a complicated and long-winded process, something the SCORE tool aims to deal with.

The SCORE tool implements the rules defined by the SCORE method, but also contributes to the risk analysis itself. Important to notice is that the tool is implemented as a prototype used to prove the realisation, validity and practical utility of the method. At the same time, the tool proves the necessity of the method being tool-supported in order for the risk analysts to willingly adopt the SCORE method due to the complexity of the latter.

To summarise; SCORE is a method for qualitative and quantitative risk estimation supported by a computerised tool integrated in the existing CORAS tool. It aims to simplify the process of risk estimation for the risk analysis team by offering an automation of risk estimation calculations through a specially designed tool interface. In short, it provides a *Simplified CORAS Risk Estimation* method, hence the *SCORE* abbreviation.

1.3 Thesis structure

The thesis is structured as follows:

Chapter 1 **Introduction**
Introduces the reader to our domain of interest and the SCORE project. It also motivates the need for and describes the contribution from the SCORE project.

Chapter 2 **Background**
Provides background material that is used as basis and/or inspiration to the development of the SCORE project. Covers *security risk analysis*, *CORAS diagrams* and *established threat and risk modelling techniques*.

Chapter 3 **Problem analysis**
Conducts a problem analysis for the SCORE project. It introduces and explains main notions and provides the problem characterisation. Furthermore, it motivates the need for the SCORE project and carefully characterises the requirements to this project in terms of success criteria.

Chapter 4 **The SCORE method**
Defines a set of rules for the behaviour of the SCORE method.

Chapter 5 **Requirements for the tool**
Introduces an overall vision of the tool and presents the requirements for the tool in terms of software requirements.

Chapter 6 **The SCORE tool**
Provides a full presentation of the SCORE tool implementation. It covers all abstraction layers from a high-level user view with live screen shots, through the architectural design and to the implementation specific details.

Chapter 7 **Evaluation**
Evaluates the SCORE method and tool based on a case study, software testing and non-empirical evaluation and discusses the liability of the results.

Chapter 8 **Fulfilment of the success criteria of SCORE project**
Validates the SCORE method and tool with respect to its defined success criteria and identified requirements.

Chapter 9 **Discussion**
Evaluates experiences and findings from the SCORE project research.

Chapter 10 **Conclusion**
Sums up the main findings and recommendations for the SCORE project and suggests future work.

References
List of references to sources of information used in the thesis.

Appendices

- The SCORE tool architectural design
- Software requirements specification (SRS)
- Security risk analysis of Stubrud sjakk-shop
- The SCORE tool test results

Chapter 2

Background

The SCORE project relates to several special fields, ranging from IT-disciplines such as software engineering, programming languages and of course computer security, to mathematical approaches to probability and fuzzy logic. In particular, the SCORE project studies security risk assessment based on CORAS that in turn applies various methods and techniques. These may be well-known and incorporated into the best practise of security risk assessment, or state of the art research under development as we speak. Anyhow; this chapter gives the reader an introduction to the background material she needs to get down to the rest of the thesis. Those already familiar with this material can skip the whole chapter or simply use it as a reference without missing any contribution from the SCORE project.

In order to understand our domain of interest, we first introduce the reader to the practise of security risk assessment. More precisely, we study CORAS security risk analysis with special focus on the security risk analysis sub-process of risk estimation.

The SCORE method builds on the established security risk analysis theories for risk estimation and applies them to CORAS diagrams. To understand CORAS diagrams sufficiently, we present the syntax, semantics and intended use of the CORAS diagrams in detail where it comprises the affected elements of the risk estimation phase.

When we develop our method, we are inspired by well-known risk analysis techniques such as FTA [5] and ETA [6] and how they use diagrams in the estimation process. We extract the core of the different notions using the ISO/IEC

14977 EBNF [7] standard. This will allow us to compare them to each other later on.

This chapter aims to introduce the reader not familiar with security risk assessment and the CORAS framework to the practise of security risk analysis. In particular, we focus on the process of risk estimation based on the CORAS method, and we provide the reader with a brief introduction to selected relevant techniques that constitutes the basis for and/or yields inspiration to the SCORE project.

2.1 Security risk analysis

1.3.14 Risk analysis:

> *"systematic process to understand the nature of and to deduce the level of risk"*
>
> AS/NZS 4360:2004 [8]

The CORAS method is to a large extent based on the worldwide acknowledged *Australian/New Zealand Standard®AS/NZS 4360:2004* for risk management [8]. The quotation above from this standard defines precisely what is the context of risk analysis. In CORAS we talk about *CORAS security risk analysis* which we abbreviate to simply *risk analysis*.

Risk analysis attempts to answer three fundamental questions [9]:

- what can go wrong (by hazard identification)
- how likely is this to happen (by frequency analysis)
- what are the consequences (by consequence analysis)

When we have the answer to these questions, we say we have identified the *risk picture*. Following AS/NZS 4360 CORAS divides the risk analysis process into 5 phases: (1) context establishment, (2) risk identification, (3) risk estimation, (4) risk evaluation and (5) treatment identification.

CORAS diagrams applies to all of them. In the context establishment, *asset diagrams* are used to specify the assets within the target in relation to their stakeholders. Another type, what we call *threat diagrams*, applies to the next two phases where risks are identified and estimated. The fourth phase uses *risk diagrams* while how risks are treated are documented in *treatment diagrams*.

2.2 Established threat and risk modelling techniques

Below we study established threat and risk modelling techniques. The chosen techniques are *fault tree analysis (FTA)*, *event tree analysis (ETA)* and *attack tree analysis (ATA)*. We base our selection of techniques on recommendations from the SINTEF risk analysis guide [6] together with investigations of the SCORE project needs.

2.2.1 Fault tree analysis (FTA)

A fault tree [5] is a logical diagram describing the relation between an unwanted incident in a system and the reasons causing the incident. The tree is a variant of an AND/OR tree with the unwanted incident as the *root* or the *top event*. The tree grows top down; each node's children creating a sub tree successively down to the leaf nodes. The leaf nodes are defined as the initiating events. It is up to the creator of the tree to decide the level of details for the leaf nodes according to the target of analysis.

Traditional fault tree analysis involves determination of *minimal cut-sets*. Minimal cut sets are all the unique combinations of component failures that can cause system failure. A qualitative analysis of the fault tree would start with the minimal cut set to identify the initiating event(s) causing the unwanted incident.

Each node in the fault tree can be labelled with numbers representing failure probability. This may be hard to obtain, but if we assume independent events and that the probabilities of the initiating events can be estimated, we could perform a quantitative analysis. A computer system could calculate the probability of each node of the minimal cut set resulting in the estimated probability of the top event. Note that each estimated probability should be given in terms of a limited time-frame. The probability of the top event would then correspond to the frequency the top event occurs.

Possible results of a fault tree analysis:

- a list of possible combinations of environmental factors, human mistakes, normal scenarios and component error leading to an unwanted incident
- the probability for unwanted incidents to occur within a certain time period (frequency)

Probability aggregation

Each node in the fault tree can be assigned a probability value. When all leaf nodes have a probability value, the fault tree can aggregate these values upwards the tree to the top.

Probability aggregation in fault trees are conducted based on the ports they aggregate over. Logical OR gates summarise the input events, while AND gates multiplies. Suppose top event te caused by event $e_1(l_1)$ to $e_n(l_n)$ with the likelihood values l_1 to l_n, respectively through an OR (2.1) and an AND (2.2) gate:

$$l_t = l_1 + l_2 + \ldots + l_n \tag{2.1}$$

$$l_t = l_1 \times l_2 \times \ldots \times l_n \tag{2.2}$$

where l_t is the aggregated probability value for top event te.

Constructs

A fault tree consists of nodes and logical ports illustrating the relations between the nodes.

The constructs [6][5]:

- Nodes:
  - root – unwanted incident
  - intermediate
  - leaf – initiating event
- Logical ports:
  - AND
  - OR

EBNF-presentation of fault tree

Syntax presentation on extended Backus-Naur form:

```
<tree>     := <node>|<node>AND<tree-set>|<node>OR<tree-set>|<node><tree>
<tree-set> := {tree}|{tree}U<tree-set>
```

Examples of FTA application

Graphical example of fault tree and the construction of it:

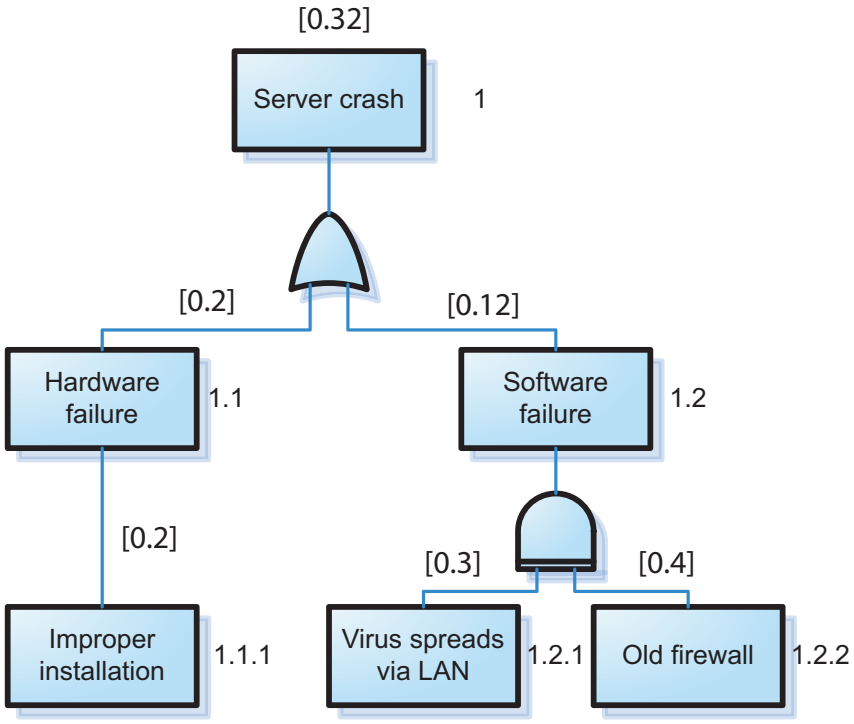

Figure 2.1: Example of construction of fault tree from BNF notation

Note that the numbers in brackets denotes probabilities while the others are only to identify each unique node and can be found in the syntactical presentation below.

Syntactical presentation of the same tree (from figure 2.1) with the notations mapped to the node instances:

```
<tree> - 1
   <node> -1
   OR
   <tree-set>
      <tree>
         <node> - 1.1
         <tree>
            <node> - 1.1.1
      U
      <tree-set>
         <tree>
            <node> - 1.2
            AND
            <tree-set>
               <tree>
                  <node> - 1.2.1
               U
               <tree-set>
                  <tree>
                     <node> - 1.2.2
```

Scope of technique

The fault tree analysis technique has shown to be most successful in risk- and reliability analysis. It was originally invented by the BOEING COMPANY in 1961 for system safety analysis [5]. A fault tree can be used for qualitative analysis, quantitative analysis or both. During a design phase, the fault tree analysis could reveal "hidden" errors caused by component faults.

2.2.2 Event tree analysis (ETA)

An event tree [6] is a logical diagram describing possible event chains caused by an undesirable initiating critical event. An event chain is a series of events caused by other events. It starts by the initiating event and results in an accident. The purpose of the technique is to identify the critical event chains and help mitigate their negative effects.

The *root* or the *top event* of the event tree is the initiating event. The tree is binary (however not strict – a node may have only one child, but never more than two) and each following event is ordered at the next level in the tree. At each event the tree splits in two according to the success or failure of the event.

Constructs

The constructs [6]:

- Nodes:
  - root – initiating event
  - intermediate
  - leaf – resulting accident
- Binary ports:
  - success
  - failure

EBNF-presentation of event tree

Syntax presentation on extended Backus-Naur form:

```
<event tree> : = <initiator><tree>
<tree>       : = <node><success><failure>
<success>    : = <tree>|<terminal>
<failure>    : = <tree>|<terminal>
```

Examples of ETA application

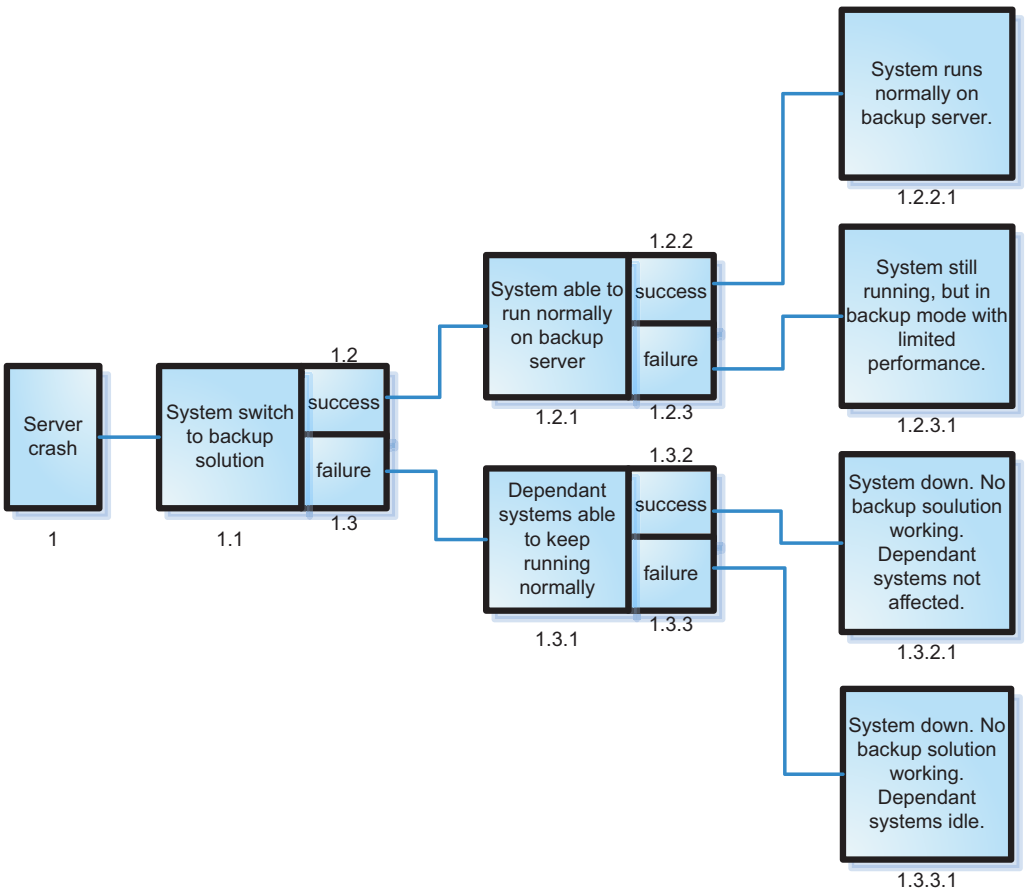

Figure 2.2: Example of construction of event tree from BNF notation

Syntactical presentation of the same tree (from figure 2.2) with the notations mapped to the node instances:

```
<event tree>
   <initiator> - 1
   <tree> -
      <node> - 1.1
      <success> - 1.2
         <tree>
            <node> - 1.2.1
            <success> - 1.2.2
               <terminal> - 1.2.2.1
            <failure> - 1.2.3
               <terminal> - 1.2.3.2
      <failure> - 1.3
         <tree>
            <node> - 1.3.1
            <success> - 1.3.2
               <terminal> - 1.3.2.1
            <failure> - 1.3.3
               <terminal> - 1.3.3.1
```

Scope of technique

An event tree analysis is a part of most reliable risk analysis's of technical systems. The technique can also be used in reliability analysis.

2.2.3 Attack tree analysis (ATA)

The attack tree analysis [10] is a methodical way of describing threats against, and countermeasures protecting a system. The technique is developed by the computer security specialist Bruce Schneier with the purpose of being used in threat modelling of computer systems. The technique provides a methodical way of representing the security of systems.

The attack tree represents attacks against a computer system in an AND/OR tree structure with the goal as the *root* node or the *top event*. The goal is the defined objective of the attack and can possibly be achieved through any minimal cut set of the tree from the leaves to the root.

The analysis of the tree is analogous to the traditional fault tree analysis. The minimal cut set consists of one or more initiating event depending of the logic of the tree. Any node can be assigned probability values for quantitative or qualitative analysis.

Constructs

The constructs [10]:

- Nodes:
  - root – unwanted incident (goal of the attack)
  - intermediate
  - leaf – initiating event (different ways of achieving the goal)
- Logical ports:
  - AND
  - OR

EBNF-presentation of attack tree

Syntax presentation on extended Backus-Naur form:

```
<tree>     := <node>|<node>AND<tree-set>|<node>OR<tree-set>|<node><tree>
<tree-set> := {tree}|{tree}U<tree-set>
```

Scope of technique

The intended use for this technique is threat modelling of computer systems. The attack tree provides abilities to make calculations about security and compare the security of different systems.

2.3 CORAS diagrams

CORAS diagrams are created using a customised modelling language. Figure 2.3 shows an example of a threat diagram with explanations of the diagram elements (*they* are of course not a part of the language). Although there are

various CORAS diagrams for different phases and purposes, we only consider threat diagrams due to the scope of the SCORE project.

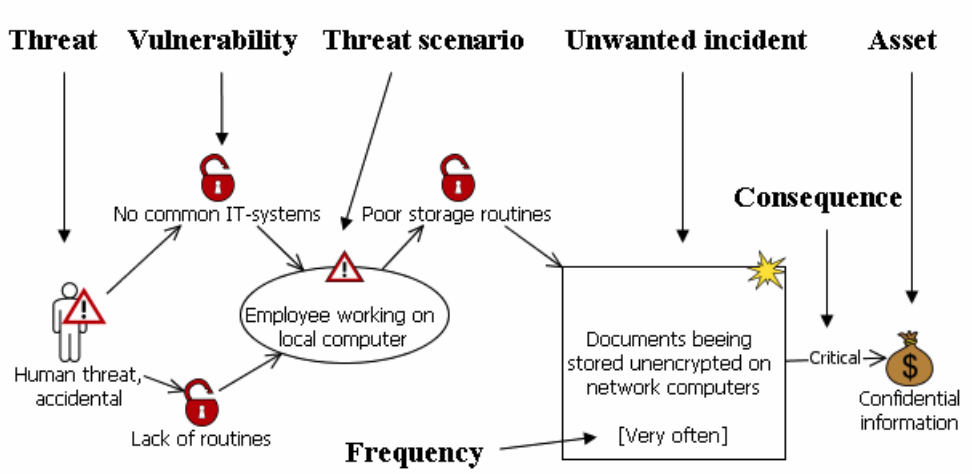

Figure 2.3: CORAS diagram example

2.3.1 Syntax

The syntax of CORAS diagrams[1] is explained three different ways. First we give a textual explanations of all the constructs (diagram elements) thereafter display them graphically. Finally we present the constructs using EBNF notation.

Constructs

Explaining the diagram elements would be a natural part of any risk analysis where there are participants present without prior knowledge to CORAS diagrams. During the risk analysis the participants will together with the risk analysis team model different CORAS diagrams using these constructs. It is of high importance that the participants know the definition of the graphical symbols in order to perform correct modelling. Figure 2.4 presents the constructs

[1] Although the SCORE tool-supported method applies to threat diagrams exclusively (hence do not bring all of the possible diagram elements into use), we have chosen to present the totality of the constructs instead of leaving just a few out. The legal constructs for threat diagrams are: *threats* (all), *vulnerability*, *threat scenario*, *unwanted incident*, *asset*, *initiate* and *impact relationship*, *likelihood*, *consequence*, *AND* and *OR*.

graphically as they are used in CORAS diagrams and table 2.1 explains each of their definitions literally [11].

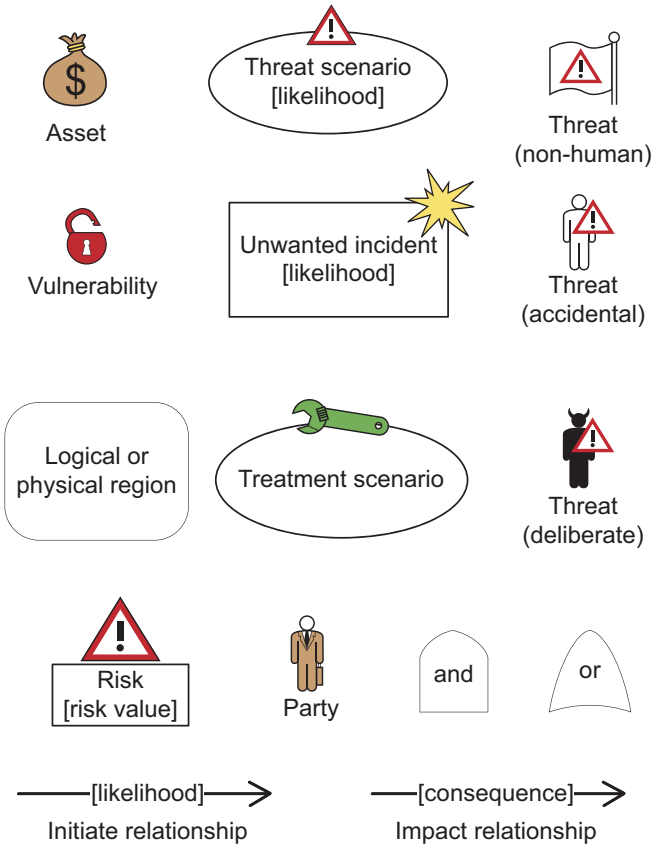

Figure 2.4: CORAS constructs

Element	Description
stakeholder	those people and organisations who may affect, be affected by, or perceive themselves to be affected by, a decision or activity regarding the target of analysis
asset	something to which a stakeholder directly assigns value and, hence, for which the stakeholder requires protection
vulnerability	weaknesses which can be exploited by one or more threats
threat	a potential cause of an unwanted incident, can be categorised as either human or non-human (environmental), human threats can also be said to have accidental or deliberate origin
unwanted incident	an event that may harm or reduce the value of assets and is something we want to prevent
risk	the chance of something happening that will have an impact upon objectives (assets), defined to consist of an unwanted incident, a likelihood measure and a consequence
likelihood	a general description of frequency or probability
consequence	damage to an asset
treatment	the selection and implementation of appropriate options for dealing with risk

Table 2.1: CORAS constructs definitions

EBNF-presentation of coras diagrams

Syntax of CORAS diagrams presented using EBNF [12] notation:

<relation> := <initiate>|<impact>;

<initiate> := <threat> $\xrightarrow{[\text{<vulnerability set>}][\text{<likelihood>}]}$ <threat scenario> |
<threat> $\xrightarrow{[\text{<vulnerability set>}][\text{<likelihood>}]}$ <unwanted incident> |
<threat scenario> $\xrightarrow{[\text{<vulnerability set>}][\text{<likelihood>}]}$ <threat scenario> |
<threat scenario> $\xrightarrow{[\text{<vulnerability set>}][\text{<likelihood>}]}$ <unwanted incident> |
<unwanted incident> $\xrightarrow{[\text{<vulnerability set>}][\text{<likelihood>}]}$ <threat scenario> |
<unwanted incident> $\xrightarrow{[\text{<vulnerability set>}][\text{<likelihood>}]}$ <unwanted incident>;

<impact> := <unwanted incident> $\xrightarrow{[\text{consequence}]}$ <asset> |
<threat scenario> $\longrightarrow$ <asset>;

<threat> := <deliberate threat> | <accidental threat> | <non-human threat>;

<deliberate threat> := identifier;

<accidental threat> := identifier;

<non-human threat> := identifier;

<vulnerability set> := {vulnerability} $^{-}$;

<vulnerability> := identifier;

<threat scenario> := identifier [(<likelihood>)];

<unwanted incident> := identifier [(<likelihood>)];

<asset> := identifier;

<likelihood> := 'Linguistic term' | 'Numerical value';

<consequence> := 'Linguistic term' | 'Numerical value';

<identifier> := 'Natural language description';

2.3.2 Pragmatics

The CORAS guideline [13] explains how to use CORAS diagrams through an example driven introduction. Threat diagrams that SCORE applies to are subject to the risk identification and risk estimation phases. The first phase identifies unwanted incidents, threat scenarios, vulnerabilities and their causing threats while the latter takes these as input and applies risk estimations.

The risk estimation phase aims to assign likelihood and consequence values to unwanted incidents. Likelihoods may be expressed as frequency values (i.e. 5 times per year) or probability values (i.e. 0.3 or 30% probability). They may either be qualitative linguistic values (such as *often*) or quantitative numerical values (as already exemplified). Numerical values are often defined in the context identification as numerical sets mapped to the linguistic expressions since they are easier to relate to and work with.

Consequence values are assigned to the relation between unwanted incidents and assets in a similar manner, but their value (qualitative or quantitative) varies with respect to the current asset under investigation and how this is measured.

There are different strategies to come up with the likelihood values for unwanted incidents. If the risk analysis participants have statistics or other knowledge of the incidents, they may be assigned directly. Often this is not the case due to the complexity of the unwanted incident and that it may be composed by a number of threat scenarios or even other unwanted incidents. The participants then try to estimate likelihoods for the related threat scenarios and the risk analysis team will use this information to combine an aggregated likelihood for the unwanted incident. When there are still problems extracting enough reliable data, the risk analysis team may suggest basing the estimations on historical data or personal experiences, or attempt to perform a fault tree analysis for precise calculations.

In cases where likelihoods are assigned to threat scenarios or unwanted incidents not directly associated with an asset in the diagrams, hence requiring some kind of computing of the likelihoods, the risk analysis team may apply *"an informal method that is quite straight forward and transparent and suitable for the brainstorming setting"* [11].

2.3.3 Semantics

A textual syntax and a structured semantics for the CORAS diagrams is precisely defined in *Structured semantics for the CORAS security risk modelling language* [14] or *the CORAS semantics* for short. The work is intended to enable users of the CORAS language to easily extract the precise meaning of a CORAS diagram. Although all five CORAS diagrams are handled in detail in the CORAS semantics, we only focus on the *threat diagram* because of the scope of the SCORE project. Below we reproduce what is relevant to the SCORE project from the CORAS semantics using the same outline as the original work.

Threat diagrams

A threat diagram presents a chain of events initiated by threats that end up in an *unwanted incident*. The unwanted events occur due to *vulnerabilities*. *How* this occurs is described by so-called *threat scenarios*. When an unwanted incident occurs, it may have consequence for the assets. Threat diagrams may apply the following constructs from 2.3.1 above: deliberate, accidental and non-human threats, vulnerabilities, threat scenarios, unwanted incidents, and assets, and the relations: initiate and impact. Threat scenarios and unwanted incidents may be assigned a likelihood value.

There are clearly defined rules for how the relations may be used and which

elements are involved. First of all, they are all binary and directed. The impact relationship may only be used to relate an unwanted incident to an asset that is harmed as a consequence of this incident. Impact relationships may be assigned a consequence value. An initiate relationship applies to all the constructs except the asset, but not to all possible combinations. A threat diagram is always initiated by a threat that exploits vulnerability, either directly to an unwanted incident or through a threat scenario. The two latter may further initiate other threat scenarios or unwanted incidents. Initiate relationships may be assigned a likelihood value.

Translation from graphical to textual syntax

The CORAS semantics defines translation rules for all possible relations. Here we present the scenario where a threat scenario initiates an unwanted incident (figure 2.5), and because the other scenarios are analogues we believe that they are easy to picture for the reader. The arrow $\tilde{\rightarrow}$ denotes the transformation from graphical to textual syntax.

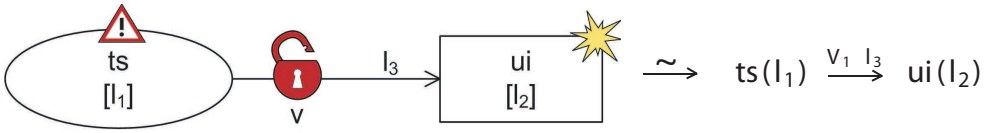

Figure 2.5: Graphical to textual translation of a threat scenario to unwanted incident initiate relationship

Structured semantics for threat diagrams

Following up the example from above, we present: *Initiate relation, threat scenario to unwanted incident* from the CORAS semantics:

$$
\begin{array}{rcl}
[\![ts(l_1) \rightarrow ui(l_2)]\!] & := & \text{After } [\![ts(l_1)]\!] \text{ has taken place, } [\![ui(l_2)]\!] \text{ may be initiated} \\
[\![ts(l_1) \xrightarrow{l_3} ui(l_2)]\!] & := & \text{After } [\![ts]\!] \text{ has taken place, there is a } [\![l_3]\!] \text{ that } [\![ui(l_2)]\!] \text{ will be initiated} \\
[\![ts(l_1) \xrightarrow{V_n} ui(l_2)]\!] & := & \text{After } [\![ts]\!] \text{ has taken place, } [\![V_n]\!] \text{ may be exploited to initiate } [\![ui(l_2)]\!] \\
[\![ts(l_1) \xrightarrow{V_n l_3} ui(l_2)]\!] & := & \text{After } [\![ts]\!] \text{ has taken place, there is a } [\![l]\!] \text{ that } [\![V_n]\!] \text{ will be exploited} \\
 & & \text{to initiate } [\![ui(l_2)]\!]
\end{array}
$$

And with the threat scenario and unwanted incident from above defined as:

$[\![ts(l_1)]\!]$:= threat scenario 'ts', which has likelihood 'l_1'
$[\![ui(l_2)]\!]$:= unwanted incident 'ui', which has likelihood 'l_2'

Chapter 3

Problem analysis

In this chapter we conduct a problem analysis for the SCORE project. The main purpose of the chapter is to characterise a problem description which the rest of the thesis will investigate. The first thing we describe in this chapter is a case study conducted with the purpose of analysing our domain of interest. The case study is an industrial CORAS security risk analysis from which experiences are utilised to obtain a problem characterisation. With the problem characterisation in hand, we investigate what are the possible theoretical approaches to our problem description and relate the background material provided in chapter 2 to the SCORE project and its problem description. Next we decide a research method that appoints the further progress of the SCORE project and assures that it follows acknowledged research processes. Based on all of the above, we finally come up with the success criteria for the SCORE project.

3.1 Aim of the problem analysis

The problem analysis is written to clarify what are the specific tasks for the project. In order to come up with useful results the problem analysis must identify the needs for the SCORE project that also yields the motivation for executing the project.

The result of the problem analysis should be a precise and accurate characterisation of the success criteria for the SCORE project. The success criteria are later used to verify whether or to what extent the SCORE project meets its requirements and fulfils its tasks.

Achievement of success is accomplished through certain methods of research and studies. The problem analysis should also describe suitable methodologies and identify what information is needed from which information sources.

3.2 Case study

The SECURIS project is an EU-funded project that builds, amongst others, on results from the completed research project CORAS and aims to establish a computerised method for the development of secure IT systems. *The eight SECURIS field trial* [4] is used as a case study to identify the needs and requirements for the SCORE project. Experiences and recommendations from the risk analysis team are documented in an own evaluation report [4]. We examine the *risk estimation phase* section of the report with particular interest below and reproduce the interesting findings.

From the section explaining the practical conduction of the risk estimation phase, we extract the following information:

> *The risk analysis team aimed to assign frequency values directly to the unwanted incidents harming an asset. Due to the complex nature of some of the unwanted incidents, the participants had to estimate values for the simpler threat scenarios or other unwanted incidents and use this information to make a roughly estimated frequency value.*

The above is accompanied with the following recommendation from the risk analysis team:

> *When the participants have problems estimating the frequency value of an unwanted incident, the risk analysis team should try to focus on the related threat scenarios of less complexity and use this information to estimate the combined frequency value.*

The problems stated above could be solved with a method for estimating the likelihood (*here:* frequency) values at the threat scenario level with the intention of aggregating a combined likelihood value for the unwanted incident based on the threat scenario input. The SCORE method should aim to offer such a feature.

From the evaluation report we find more valuable information, now concerning the live conduction of the risk analysis and how clear and easy-to-understand the risk estimation phase and diagrams appear:

> *On complex diagrams with multiple paths towards an asset there was confusion of what was focus at the present moment.*

And the evaluation report further recommends on basis of the experiences:

> *The risk analysis team recommends using the CORAS tool live during the risk analysis workshops with well prepared diagrams. To provide the best overview and easy to follow diagrams, they should be kept small enough for a single projected screen size, but yet in context to retain proper overview.*

This implies that the phase of risk estimation may be confusing to the risk analysis participants. We believe that the SCORE tool designed to simplify how risk estimation documentation is retrieved and processed and with the possibility to extract one single path from the rest of the diagram, would help the participants getting a more perspicuous overview of the risk estimation phase.

3.3 Problem characterisation

The problem characterisation consists of three coherent parts: an *overall problem description*, a *motivation* and a *refined problem characterisation*. The first section defines an overall hypothesis we wish to investigate in this thesis. Next we motivate our choice of problem area and explain the needs related to the current research area. Finally we refine the overall hypothesis into predictions that can be used for validation.

3.3.1 Overall problem description

Our research is subject to the field of *technology research* (which is explained and motivated in section 3.5) and will therefore yield an overall hypothesis proposing the statement: *an artefact satisfying a need* [15]. The artefact this that respect, is the SCORE tool-supported method.

Overall hypothesis

H1: *a carefully designed tool-supported method will increase the efficiency of risk estimation using CORAS diagrams*

CORAS diagrams do currently not possess any structured methodical way to support the risk estimation phase of a CORAS risk analysis. Risk estimation

is dealt with in a manual matter left up to the field experts. If we apply a tool-supported method to this phase, we believe that the conduction of the risk estimation phase will perform easier and the results become more accurate. In order to investigate *efficiency*, we need to look at both effectiveness in a time related fashion and correctness related to the result of likelihood calculations.

Our hypothesis immediately raises two questions: *How can risk estimation be automated with a computerised tool? What rules shall such method build on?* In prolongation to the overall question we will investigate if and possibly how the established techniques from chapter 2 may be applied to CORAS diagrams in order to answer the questions.

3.3.2 Motivation

Our problem description is mainly motivated by the need we reveal for our artefact in section 3.2. The study shows that the problem is highly relevant because it may cause unnecessary confusion during a risk analysis [4] (examined in section 3.2 of this chapter). Although the problem has obviously been addressed previously in the research of CORAS diagrams, no formal method has yet been evolved. We believe that the SCORE tool-supported method contribute to CORAS by increasing the efficiency of CORAS risk estimation.

In a scientific research relation, our problem has proven to be of great interest. CORAS as a research project has demonstrated scientific significance and topicality by being an EU-funded research project (completed in September 2003) [16] in the first place. The fact that CORAS is followed up by SECURIS [17] the next year (also EU-funded) that in turn is succeeded by DIGIT [18] this year that both continue pursue the development of CORAS, consolidates the scientific significance of CORAS.

Furthermore, problems about decision making methods based on fuzzy or qualitative data (as we experience during risk estimation of CORAS risk analysis) has proven to be of scientific interest by a number of scientists and their work [19][20][21].

3.3.3 Refined problem description

In order to be able to test the overall hypothesis **H1**, it needs to be refined into a set of specialised predictions that assemble the overall success criteria for the SCORE project. The predictions constitutes the basis for the hypothesis testing,

hence they need to be expressed in such a way that they are easily rejectable (falsification).

Before we propose the predictions we need to explain a complicated but rather important (in terms of hypothesis testing) phrase from **H1**, namely *increased efficiency*. Increase is defined as addition or enlargement in size, extent, or quantity [22] while efficiency considers *effectiveness* in terms of time-saving and added quality in our context.

In order for a risk estimation to become more accurate with respect to the likelihood values, we must compare it to what is the current situation. CORAS diagram pragmatics (section 2.3.2) proves that all estimations of likelihoods are performed using human expertise and experience. Weaknesses from thence are related to human limitations such as dealing with large or complicated data and intricate relations. Probability theories prove that humans have the tendency to overestimate the likelihood of conjunctive events, and especially when the events are complex or compound [23], as CORAS unwanted incidents may be.

In order for a tool-supported method to improve this method and deal with the problem, it must fulfil the overall success criteria we claim as predictions below. These will be further specialised into detailed success criteria at the end of this chapter.

A carefully designed tool-supported method:

Overall success criterion 1

P1: *will provide the ability to merely assign likelihood values to threat scenarios and automate likelihood aggregation to unwanted incidents*

Overall success criterion 2

P2: *will be able to calculate likelihood values for diagram elements initiated through intricate relations*

Overall success criterion 3

P3: *will handle an arbitrary large number of input*

When concerning effectiveness in terms of time-saving, the tool-supported method should apply means to what type of input to process.

Overall success criterion 4

P4: *will provide the ability to merely deal with qualitative data, but still yield calculations thereupon*

Overall success criterion 5

P5: *will provide the ability to merely assign likelihood values to threat scenarios and automate likelihood aggregation to unwanted incidents*

3.4 Discussion

In this section we discuss whether, or to what extent, the analysis techniques described in chapter 2 can be performed with CORAS diagrams. First we evaluate the analysis techniques FTA, ETA and ATA against CORAS diagrams, and based on this follows a description on how to develop the SCORE tool-supported method for risk estimation.

When we talk about CORAS diagrams in this chapter, we refer to CORAS threat diagrams exclusively.

3.4.1 Established modelling techniques and the needs of CORAS diagrams

The main problem to be addressed here is *if* and possibly *how* the established analysis techniques (fault tree, event tree and attack tree analysis from chapter 2) can be conducted using CORAS diagrams.

The attack tree is not discussed alone, but is considered as a specialisation of the more general fault tree. Their syntax is similar, but the attack tree specifically turns upon security of computer systems. Since SCORE concerns analysis of computer systems, it would be more appropriate to consider the attack tree analysis as the variant of performing a fault tree analysis. We therefore refer to the fault and attack tree analysis as one but use the fault tree notation due to its common industrial application, hereby FTA.

FTA is heavily incorporated in industrial risk analyses and has ever since the original development of CORAS been a contemplated part of its framework [16]. The intention was that CORAS diagrams can be used to construct fault trees. ETA is mentioned in the manual for the CORAS methodology [24] found in the CORAS

tool as an example of conventional risk analysis methods that could possibly attend a risk analysis together with Hazard and Operability Analysis (HazOp) [6], Fault Tree Analysis (FTA) and Failure Mode, Effects, and Criticality Analysis (FMECA) [6]. Although the the CORAS methodology manual is currently outdated (refer to [13] instead), Hogganvik and Stølen claim FTA, ATA and ETA to be supported in their graphical approach [25] originated from the CORAS project, thus should be facilitated by the CORAS tool. HazOp and FMECA are beyond the scope of this report, but FTA and ETA are discussed exhaustively later.

Fault trees and event trees both have a similar, yet different, characteristic together with CORAS diagrams. Threat diagrams [13] have a much similar structure to fault trees. An unwanted incident (or even a threat scenario in some cases) can be thought of as the top event of a fault tree with a random number of threat scenarios describing how the unwanted incident may occur. CORAS diagrams starts with a threat as the top event which is the leaf node in a fault tree, hence they grow in opposite directions. Event trees however, concur to CORAS diagrams with respect to reading direction. They start with an initiating event that could be some identified threat, and follows the chain of events leading to the accident (or unwanted incident). Below we look into each of them in detail.

FTA in CORAS diagrams

To create a fault tree we need to identify the unwanted incident as the top event. Then we can successively construct the tree by identifying possible incidents that may cause the top event. When all incidents are identified, we see the actual use for this technique. Likelihood values may be assigned to each of them yielding:

> *after incident* A *has taken place, there is a likelihood* l *that incident* B *will be initiated*

where A and B are nodes in the tree and l is assigned the relation between them.

With likelihood values assigned to a *minimal cut-set* (refer to chapter 2.2.1) or more of the incidents, the fault tree aggregates the likelihood up to the top event.

To perform an FTA in a CORAS diagram we would choose an unwanted incident (or a threat scenario) to represent the top event we wish to investigate. We say "or a threat scenario" because we could just as well investigate a threat scenario as an unwanted incident even though the latter are most common and what we aim for in the end. The initiating threat scenarios (or unwanted incidents) can

easily be translated into the internal nodes of the fault tree as the children of the top event. Again; unwanted incidents in parenthesis means that they could also be the initiator though this is more seldom.

When it comes to selecting the leaf nodes for the fault tree, the CORAS elements are not merely intuitive. An FTA defines the leaf nodes as the initiating events which would imply using the threats from CORAS diagrams. Fault trees need to assign likelihood values to the relations *from* the leaf nodes as input for the likelihood calculation. This ability is documented in the CORAS semantics [14] and yields to all initiating relationships, including those from threats. However, this is not best practise from CORAS risk analyses where we normally applies likelihood values to the first threat scenarios initiated *by* the threats.

Allowed mappings:

FTA	**CORAS diagram**		
root		threat scenario	**unwanted incident**
intermediate		**threat scenario**	unwanted incident
leaf	threat	**threat scenario**	unwanted incident

Table 3.1: Mappings from fault tree to CORAS diagram

where the preferred (and most intuitive) CORAS elements to their respective tree construct are in boldface.

The missing FTA constructs we have not discussed yet are the logical ports AND and OR. The CORAS semantics does not define the logical ports as a part of the language even though we have presented them as a part of the CORAS constructs in chapter 2.3. If we look into an initiate relationship and picture two or more initiating nodes, we are likely to interpret them as independent when nothing else is claimed. Given this assumption, we would calculate the likelihood values using an OR port.

Otherwise, if we must expect dependencies between nodes, this should be explicitly illustrated using the AND port. Note that use of the AND port should be deliberate and denote the situation where *all* input nodes (to the AND port) are interrelated and must happen before in order for the subsequent to be initiated. The likelihood of the latter would hence be the likelihood for this to occur [26].

Other possible more diffuse dependencies would most certainly affect possible likelihood calculations and require research in special fields of dependability,

thus fall outside of the scope of the SCORE method.

ETA in CORAS diagrams

An event tree presents a dynamic view of a scenario caused by an initiating event leading to an accident. A CORAS diagram would be able to capture the sequential event-flow if we apply a threat or vulnerability as the top event of the event tree and an unwanted incident as the accident. Similar to the previous discussion on FTA, the flexibility of the CORAS elements allows us to model different elements as different nodes, but clear preferences from the CORAS risk analysis method yield the boldface ones.

Allowed mappings are:

ETA	**CORAS diagram**		
root	**threat**	threat scenario	unwanted incident
intermediate		**threat scenario**	unwanted incident
leaf		threat scenario	**unwanted incident**

Table 3.2: Mappings from event tree to CORAS diagram

where the preferred (and most intuitive) CORAS elements to their respective tree construct are in boldface.

Similar to fault trees also event trees have some differences from CORAS diagrams. First of all event trees do not allow more than one root node. Secondly there are no (binary) constructs for *success* or *failure*. However, CORAS diagrams have likelihood values for the relationships to each node defined as the likelihood for the node to initiate another. In such manner, the ETA success would mean likelihood 1 and failure 0.

3.4.2 Developing the SCORE method for risk estimation

When we develop our method there are several aspects we need to consider in addition to the mappings we have suggested above from the established modelling techniques. First of all we need to relate the techniques to both a quantitative and a qualitative approach. Fault trees have clearly defined rules for how to calculate the likelihood values according to the composition of the node's children and their relations through either logical AND or OR ports.

CORAS diagrams should define unique rules for every possible relation in a similar fashion used in FTA.

More problems arise when we only have qualitative data as input. Therefore we need to create intervals for the qualitative information and categorise the statements. Further, specially designed rules to deal with intervals must be defined explicitly.

Another aspect related to qualitative input is how to handle such from an arbitrary number of sources. Multiple statements may support or contradict one another. Supporting statements would get a more reliable assertion the more statements we gain. If the multiple statements spreads out or contradicts each other, we may possibly create intervals in which the assertion is likely to be true. The question is whether or not to employ this information into the CORAS diagrams. This could be accomplished with a reliability value following each node in the tree, or the value could be baked into the already existing probability value. More likely would be to apply theories for multiple criteria decision making for fuzzy data [19], and the benefit would be that one single statement could not possibly bias the result. The drawback is that we may not have a sufficiently large number of statements available to make a certain statement do impact on the result and the complexity of such method would be tremendous.

3.4.3 Developing the SCORE tool supporting the SCORE method

The SCORE tool must be developed with the main objective being to support the SCORE method. This means that the tool must automate the risk estimation calculations defined in the SCORE method. This in turn implies that the tool must provide an interface to collect the risk estimation data. Both quantitative and qualitative input must be accounted for and processed accordingly. All rules that the method provides should be supported.

During risk estimation in a CORAS risk analysis the CORAS tool is often used live at the risk identification workshop [13]. The tool may be projected on a whiteboard or equivalent for the risk analysis participants to follow and perform the risk modelling on-the-fly. In most cases the CORAS tool and computerised modelling tools in general are unfamiliar to the participants on the client side. Much research has been put into the development of CORAS to make the execution perform as efficiently as possible [25] both with respect to method and tool. The SCORE tool should of course aim to continue pursue this mentality. Motivated by this, we believe that the SCORE tool shall be integrated with the existing CORAS tool as much as possible. If it acts as a stand-alone module working in

parallel with the CORAS tool instead, it could possibly result in extra confusion for the risk analysis participants.

Software development may be an endless process; functionality and usability features may be added and/or improved and user requests may exceed all reasonable limits given the available resources. The purpose for the SCORE tool implementation is to develop a functional prototype that provides enough functionality that are required for validation of the SCORE method and tool [15]. What is enough functionality is defined in the success criteria (section 3.6) based on the discussion here and above. However, a full requirement specification document (provided in appendix B) should be developed for eventual further implementation of the SCORE tool.

3.5 Research strategy

This section explains how the SCORE project is conducted in practise and what type of research strategy the project follows throughout its life. First we introduce a formal procedure for technology research [15] with a particular interest in software engineering [27]. Then we argue the concerns for SCORE before proposing the selected research strategy.

The SCORE project will develop both a theoretical method and a computer software prototype. In that concern, this type of research belongs to the special field of *technology research* further specialised as *software engineering* and is subject to certain practises and guidelines.

3.5.1 Software engineering and technology research

Technology research is motivated by the need for developing a new or improving an existing artefact. The artefact in this case would be the SCORE tool-supported method containing both a method for risk estimation and the computerised tool supporting this. The need for the artefact is substantiated through a problem analysis by acquiring requests and requirements for the artefact from existing users based on field trial(s). The requirements constitute the basis for a set of success criteria upon which the realised artefact will be evaluated.

After the success criteria are created, the project moves into an innovative phase. The SCORE project needs both to come up with a *new* method for risk estimation using CORAS diagrams and to *improve* the existing version of the CORAS tool by adding a new module. Both parts are executed more or less in parallel and

follow a similar research strategy. This includes using the common practises for software engineering.

A peculiar characteristic for software engineering is the influence of human behaviour through the people developing software [27]. The SCORE project should therefore employ some of the methods for empirical research together with theoretical research strategies (described in section 3.5.2 below). The aim of the innovative phase is to develop an artefact that fulfils the success criteria. In software engineering this means to first create the requirements specifications based on the result of the problem analysis. Good and detailed specifications simplify the implementation of the system.

To prove that the artefact fulfils the requirements the system is evaluated with respect to the success criteria. In software engineering this process commences preferably already during requirements engineering designated as an iterative process. The SCORE project should result in a prototype possessing enough functionality for evaluation. In order to conduct a complete evaluation, the evaluation techniques should cover as much characteristics of the system as possible. These are proposed below.

3.5.2 Research strategy techniques

When gathering research evidence, we try to maximise three characteristics: *generalisability*, *precision* and *realism*. In formal technology research there exists certain strategies, but none of them covers all characteristics. The contingency is to choose multiple techniques that complement each other [15]. We base our selection on McGrath's eight defined research strategies [28]. The SCORE project should employ:

Initial case study – this is an initial field trial of the existing CORAS tool used in an industrial security risk analysis to identify the requirements for the SCORE tool-supported method.

Non-empirical evidence – this is the general theoretic research strategy for modelling the universal behaviour of the system. In SCORE we refer to the formal argumentation in creating the method for risk estimation together with a theoretical walk-through of the method's underlying theories.

Laboratory experiment – an experimental study of the working system with purpose to manipulate desired variables to possibly provoke unexpected behaviour[27].

Experimental simulation – the SCORE tool-supported method should be tested on a real security risk analysis with emphasis on the risk estimation phase as a test case.

The chosen techniques fulfil the three characteristics: *non-empirical evidence* applies generalisability, but lacks precision where *laboratory experiment* on the contrary covers fully up. An *experimental simulation* covers the missing one – realism.

3.6 Detailed success criteria

The success criteria assemble our expectations for the SCORE project and are used to validate the results of the project in chapter 7.

3.6.1 Success criteria for the SCORE method

The SCORE method shall: provide rules that defines the risk estimation calculations in CORAS diagrams

1. define rules for likelihood calculations on 1-to-1 initiate relationships
   - calculation of numerical values shall be supported
   - calculation of qualitative values shall be supported
2. define rules for likelihood calculations on many-to-1 initiate relationships
   - calculation of numerical values shall be supported
   - calculation of qualitative values shall be supported
3. define how to translate qualitative values into numerical sets applicable for likelihood calculations
4. define rules for dealing with inconsistencies

3.6.2 Success criteria for the SCORE tool

The SCORE tool shall: provide a tool-support for the SCORE method

1. provide the ability to assign likelihood values to threat scenarios
   - both numerical and qualitative values shall be available to be assigned

- the likelihood values shall be stored in the unique node

2. provide the ability to assign likelihood values to unwanted incidents
   - both numerical and qualitative values shall be available to be assigned
   - the likelihood values shall be stored in the unique node
3. provide the ability to assign likelihood values to initiate relationships
   - the likelihood values shall be probability values in the range [0-1]
   - the likelihood values shall be stored in the unique relationship
4. provide the ability to assign consequence values to harm relationships
5. automatically calculate possible likelihood values according to the SCORE method
   - the tool shall perform a test to validate that sufficient input data is available upon execution
   - the tool shall keep the exact calculated values for further calculations in cases of round-offs and translation between numerical and qualitative values
6. provide an overview of the properties of the diagram elements

3.6.3 Success criteria for the SCORE tool-supported method integrated with CORAS

The SCORE tool-supported method shall: be integrated with CORAS

1. apply to CORAS diagrams
   - the tool-supported method shall apply to the risk estimation phase and threat diagrams in special
2. be integrated with the CORAS tool

All of the detailed success criteria defined above originates from the overall success criteria defined under the problem characterisation in section 3.3.3. The latter depends on the detailed success criteria to be validated in order to be validated themselves; and consequently a complete validation of the detailed success criteria hence implies validation of the overall correspondingly.

Chapter 4

The SCORE method

The risk estimation phase of a CORAS security risk analysis is performed with very much of the estimation work left open to the risk analysis participants. They have total freedom in assigning likelihood values to *threat scenarios* and *unwanted incidents* in a subjective manner, regardless of current dependencies or previously defined estimation levels. This may lead to inconsistencies in the CORAS diagrams and to estimated risk levels that do not account for history or external influence.

The SCORE method aims to structure the risk estimation process and provide a defined risk estimation method constituted of a set of rules. By standardising and automating the method, the SCORE method will provide the risk estimation activity sufficient overview to avoid confusion about how to estimate an appropriate risk level for a composite diagram element.

The SCORE method bases its work first of all on the CORAS semantics [14] that defines the language of CORAS diagrams. To come up with the rules, known techniques for *Probability theory* [29] and *Set theory* [29] have been used.

This chapter describes the SCORE method. First we present an example model that will be used for exemplification throughout the chapter. When we present the rules that constitute the method, we distinguish cases dealing with quantitative data from qualitative. We handle the quantitative rules first due to their precise nature and base the qualitative rules on the quantitative ones.

4.1 Legend

Figure 4.1 describes an example scenario that we use to demonstrate the SCORE method as we define the different rules.

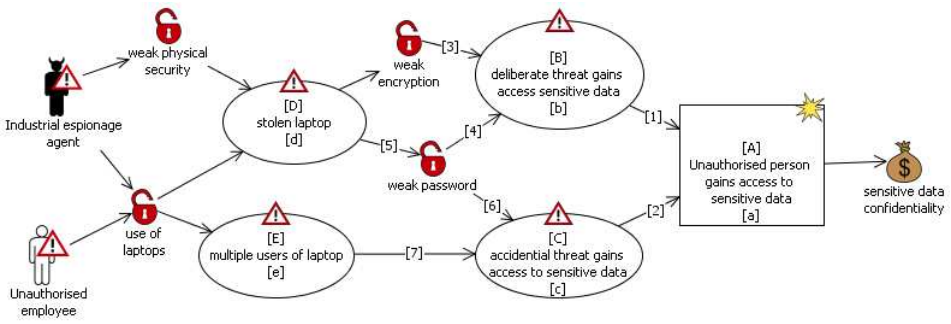

Figure 4.1: Example model for the SCORE method

To make the rules that may have a rather mathematical appearance easier to follow, the reader should be aware of the following notation:

Syntactic variables:

- *ts* – threat scenario
- *ui* – unwanted incident
- *l* – likelihood value ($l \in \{0, 1\}$) – used for annotating a node with probability
- *p* – probability value ($p \in \{0, 1\}$) – used for annotating a relationship with probability
- *n*, *m* – node n in the CORAS diagram, also called a diagram element
- *i,j,k,m* – incremental counters

Note the difference in role between the syntactic variables *l* and *p*, both ranging over probabilities. We use this convention to improve readability.

Operators:

- $\mathcal{P}()$ – "Power set" operator; the set of all subsets
- $\mathcal{L}()$ – Probabilistic domain definition
- $\mathcal{U}()$ – Linguistic domain definition
- sup – "Supremum" or least upper bound of a set
- inf – "Infimum" or greatest lower bound of a set

Vulnerabilities may be a part of any relation handled, but are omitted in this chapter since they bring no extra information to the calculation of likelihoods.

All rules in this section are presented using a simple frame consisting of two compartments and a header. The second compartment defines the rule itself based on the assumptions stated in compartment one. After the rule follows the application domain, which defines what other relationships or situations the rule applies to. Finally the rule is demonstrated with an example from the figure 4.1 above where we apply a pattern matching of the current rule elements respective to the relevant diagram notations. In other words: we substitute the rule elements with the appropriate diagram elements, and we use an arrow $\rightarrow$, not to be confused with the one used in the rules, to declare this.

The rules defined in this chapter apply only to initiate relationships. Still the relations *from* a threat or relationships *initiated* by a threat are exceptions from the rules since the CORAS semantics does not allow a threat to be assigned a likelihood value.

4.2 Quantitative rules

The first set of rules is given on the basis of exact quantitative input of probability estimations.

4.2.1 1-to-1 complete relationships

The simplest form of relationship is the 1-to-1 relationship between two diagram elements. To calculate the likelihood value of the initiated element, we apply the rule:

[1 − to − 1 complete]

$n_1(l_1) \xrightarrow{p} n$

$n_1(l_1) \xrightarrow{p} n(l)$
where
$l = l_1 \times p$

Given the likelihood value l_1 of the initiating element n_1 and the probability p for n_1 to initiate n, we may calculate the likelihood l of n with $l = l_1 \times p$. This can be done by applying *probability theory of independent events* that allows us to multiply the probability and likelihood value with the desired likelihood value as result.

Application domain

$$ts_1(l_1) \xrightarrow{p} ts_2(l)$$
$$ts(l_1) \xrightarrow{p} ui(l)$$
$$ui(l_1) \xrightarrow{p} ts(l)$$
$$ui(l_1) \xrightarrow{p} ui(l)$$

Consider the relation between B and A from figure 4.1 as an example of how the rule may be applied to a CORAS diagram.

Example 4.2.1

$$ui(l) \rightarrow A(a)$$
$$ts(l_1) \rightarrow B(a)$$
$$p \rightarrow 1$$

$$a = b \times 1$$

The likelihood value a is then calculated from b and 1 multiplied.

4.2.2 Many-to-1 complete relationships

When $m \geq 2$ elements (may) initiate an element $m+1$, we say we have a *many-to-1 complete relationship* where we calculate the combined likelihood value according to the rule:

[Many − to − 1 complete]

$$n_1(l_1) \xrightarrow{p_1} n_{m+1}$$
$$\vdots$$
$$n_m(l_m) \xrightarrow{p_m} n_{m+1}$$

$$n_1(l_1) \xrightarrow{p_1} n_{m+1}(l)$$
$$\vdots$$
$$n_m(l_m) \xrightarrow{p_m} n_{m+1}(l)$$
where
$$l = (l_1 \times p_1) + \ldots + (l_m \times p_m)$$

As long as we assume all the elements $n_1 - n_m$ to be independent, the combined likelihood value l of n_{m+1} is calculated as a sum of all contributions from the initiate relationships initiated by $n_1 - n_m$.

Application domain

$ts_1(l_m) \xrightarrow{p_m} ts_2(l)$
$ts(l_m) \xrightarrow{p_m} ui(l)$
$ui(l_m) \xrightarrow{p_m} ts(l)$
$ui(l_m) \xrightarrow{p_m} ui(l)$

Consider the unwanted incident A with an arbitrary number of threat scenarios initiating A from figure 4.1 as an example of how to apply the rule to a CORAS diagram:

Example 4.2.2

$ui(l) \rightarrow A(a)$
$ts(l_1) \rightarrow B(b)$
$p_1 \rightarrow 1$
$ts_m(l_m) \rightarrow C(c)$
$p_m \rightarrow 2$

$a = (b \times 1) + \ldots + (c \times 2)$

4.2.3 Incomplete relationships

When dealing with risk assessment, we may not capture the whole risk picture; hence there may be unknown incidents we have not taken into account. The following rule aims to account for unknown causes of diagram element n:

[1 − to − 1 incomplete]

$n_1(l_1) \xrightarrow{p} n$

$n_1(l_1) \xrightarrow{p} n(l)$
where
$l \geq (l_1 \times p)$

[Many − to − 1 incomplete]

$$n_1(l_1) \xrightarrow{p_1} n_{m+1}$$
$$\vdots$$
$$n_m(l_m) \xrightarrow{p_m} n_{m+1}$$

$$n_1(l_1) \xrightarrow{p_1} n_{m+1}(l)$$
$$\vdots$$
$$n_m(l_m) \xrightarrow{p_m} n_{m+1}(l)$$
where
$$l \geq (l_1 \times p_1) + \ldots + (l_m \times p_m)$$

The application domain for both preceding rules is the same as the corresponding rules for completeness (rule 4.2.1 and 4.2.2).

4.2.4 1-to-1 complete paths

This rule defines the least set of required values for l_i and p_i in order to calculate l.

[1 − to − 1 complete path]

$$0 \leq i \leq m$$
$$n_0(l_0) \xrightarrow{p_0} n_1 \xrightarrow{p_1} \ldots \xrightarrow{p_{m-2}} n_{m-1} \xrightarrow{p_{m-1}} n_m$$

$$n_0(l_0) \xrightarrow{p_0} n_1(l_1) \xrightarrow{p_1} \ldots \xrightarrow{p_{m-2}} n_{m-1}(l_{m-1}) \xrightarrow{p_{m-1}} n_m(l)$$
where
$$l_{i+1} = (l_i \times p_i)$$
$$l = l_0 \times p_0 \times p_1 \times \ldots \times p_{m-1}$$

Yields the following *least set required (LSR)* rule with respect to l_i and p_i in order to calculate l:

[Least set required (LSR)]

$$\{l_0\}$$
$$\{p_0, p_1, \ldots, p_{m-1}\}$$

Given a l_0, we only need the set $p_0 - p_{m-1}$ (i.e. no other values for l_i required) in order to calculate l.

Application domain

$$ts_1(l_i) \xrightarrow{p_i} \ldots \xrightarrow{p_{m-1}} ts_m(l)$$
$$ts(l_i) \xrightarrow{p_i} \ldots \xrightarrow{p_{m-1}} ui(l)$$
$$ui(l_i) \xrightarrow{p_i} \ldots \xrightarrow{p_{m-1}} ts(l)$$
$$ui_i(l_i) \xrightarrow{p_i} \ldots \xrightarrow{p_{m-1}} ui_m(l)$$

The example below considers any sequence of elements *ts* or *ui*, or both combined with their respective likelihood and probability values.

Example 4.2.4

$$ui(l) \rightarrow A(a)$$
$$ts(l_1) \rightarrow E(e)$$
$$p_1 \rightarrow 7$$
$$p_m \rightarrow 2$$

$$a = (e \times 7) \times 2$$

4.3 Qualitative rules

When performing a security risk analysis, the analysis participants may find it easier to operate with qualitative terms for likelihood estimations than exact quantitative numbers. In order to perform the calculations from the previous section (4.2), we need to map the qualitative values to intervals. Hence, we also need to start off by making rules for how to summarise and multiply intervals and apply these to our findings from the quantitative rules. The probabilistic intervals in turn are defined below as sets ranging from 0 to 1.

4.3.1 Defining a set

The SCORE method defines an interval as a set of probability values. The set ranges from 0 to 1 where 0 means will *not* happen and 1 means *will* happen. We define our type p_m with the *power set* operator of the set S which is the set of all subsets:

Power set

type $\mathcal{P}(S)$
where
$\mathcal{P}(S) \stackrel{def}{=} \{T \mid T \subseteq S \wedge T \neq \varnothing\}$

Using the *power set* we can easily define a probabilistic domain with values between 0 and 1:

Probabilistic domain

type $\mathcal{L}$
where
$\mathcal{L} \stackrel{def}{=} \mathcal{P}[0 \ldots 1]$

To represent an interval in the probabilistic domain, we use a set of linguistic expression defined as:

Linguistic domain

type $\mathcal{U}$
where
$\mathcal{U} \stackrel{def}{=} \{l \in \mathcal{P}(\mathcal{L}) \mid \forall\, i, j \in l : i \cap j \neq \varnothing \Rightarrow j = i \wedge \bigcup_{i \in l} i = [0 \ldots 1]\}$

meaning that every distinct linguistic expression is unique (i.e. no overlap) and their total union constitute the full probabilistic domain $[0 \ldots 1]$.

4.3.2 Set multiplication

This rule defines multiplication of two or more sets. Set multiplication is associative the same way as normal multiplication and therefore we only need to define multiplication of two sets and not an arbitrary number of sets.

[Set multiplication]

$p_1, p_2 : \mathcal{P}([0, \ldots, 1])$

$p_1 \times p_2 \stackrel{def}{=} \{t_1 \times t_2 \mid t_1 \in p_1 \wedge t_2 \in p_2\}$

The rule yields that multiplication of two sets results in a new set where all containing elements are calculations of each of the set elements multiplied by each of the other set elements.

4.3.3 Set addition

This rule defines addition of two or more sets. Set addition is associative the same way as normal addition and therefore we only need to define addition of two sets and not an arbitrary number of sets.

[Set addition :]

$p_1, p_2 : \mathcal{P}([0, \ldots, 1])$

$p_1 + p_2 \stackrel{def}{=} \{t_1 + t_2 \mid t_1 \in p_1 \wedge t_2 \in p_2\} \cap [0, \ldots, 1]$

The rule yields that addition of two sets results in a new set where all containing elements are calculations of each of the set elements added by each of the other set elements. However we must take into consideration that the calculated set may reach beyond its upper limit 1 and round off to 1.

4.3.4 Selecting the correct interval

Below we define a rule that yields the correct behaviour of interval selection in scenarios where a calculated likelihood set i needs to be mapped to a linguistic variable. P are the predefined legal intervals.

[Interval selection]

$P : \mathcal{U}$

$i : \mathcal{P}[0 \ldots 1]$

$i \diamond P \stackrel{def}{=} p \quad \text{where} \quad p \in P \wedge \frac{sup(i) - inf(i)}{2} \in p$

4.3.5 Inconsistencies

When a diagram element occurs in different diagrams or in different, independent places in the same diagram, it may lead to inconsistencies with respect to the likelihood values. If all likelihood values for the initiated element are calculated automatically, we get the following rule:

[Inconsistencies automated]

$$n_1^1(l_1^1) \xrightarrow{p_1^1} \ldots \xrightarrow{p_{k_1}^1} m(l_1)$$
$$\vdots$$
$$n_1^i(l_1^i) \xrightarrow{p_1^i} \ldots \xrightarrow{p_{k_i}^i} m(l_i)$$

$$n_1^1(l_1^1) \xrightarrow{p_1^1} \ldots \xrightarrow{p_{k_1}^1} m(\bigcup_{j=1}^{i} l_j)$$
$$\vdots$$
$$n_1^i(l_1^i) \xrightarrow{p_1^i} \ldots \xrightarrow{p_{k_i}^i} m(\bigcup_{j=1}^{i} l_j)$$

This can be compared with a normal 1-to-many situation (section 4.2.2) and the likelihoods are simply summarised. On the other hand, when one or more of the likelihood values are inserted manually based on an expert judgement, we get another rule:

[Inconsistencies manual]

$$n_1^1(l_1^1) \xrightarrow{p_1^1} \ldots \xrightarrow{p_{k_1}^1} m(l_1)$$
$$\vdots$$
$$n_1^i(l_1^i) \xrightarrow{p_1^i} \ldots \xrightarrow{p_{k_i}^i} m(l_i)$$
$$l_1 \cap \ldots \cap l_i \neq \varnothing$$

$$n_1^1(l_1^1) \xrightarrow{p_1^1} \ldots \xrightarrow{p_{k_1}^1} m(\bigcap_{j=1}^{i} l_j)$$
$$\vdots$$
$$n_1^i(l_1^i) \xrightarrow{p_1^i} \ldots \xrightarrow{p_{k_i}^i} m(\bigcap_{j=1}^{i} l_j)$$

Since we rely more to expert judgements than calculated values, the manual values shall be weighted more significance while curtailing the sample space of $[l_1 - l_i]$. However, the expert judgements may be vague, and the calculated values being the reducing factor. In both situations we use the intersection to define the combined likelihood value.

Chapter 5

Requirements for the tool

IT-projects most often start with the client and the contractor agreeing on what shall be the delivery of the project. In system development projects the two parties normally come up with a detailed software requirements specification (SRS) document that forms a special type of contract. The SRS is used as a tool for the contractor during the system development as an instruction for what to develop. However, after delivery the client may use the SRS to ensure that the system fulfils its mission. Likewise in the SCORE project; we use the SRS to specify the SCORE tool and to compare the tool and the SRS in the evaluation phase.

The first part of this chapter introduces an overall vision of the tool and motivates the needs for the tool. This can be read by anyone. Next we present the requirements for the tool which are addressed to readers with experiences from requirements engineering or detailed knowledge of the CORAS tool.

5.1 The vision

The overall aim for the tool is to support the SCORE method through a computerised user interface. The interface provides the user with a simple way to collect risk estimation data and automatically processes it according to the method. Figure 5.1 shows what the interface looks like, and those already familiar with the already existing CORAS tool will see that the SCORE tool is in fact simply an extension of this, because the only visual evidence of the SCORE tool is the bottom tabbed panel containing the property view amongst others.

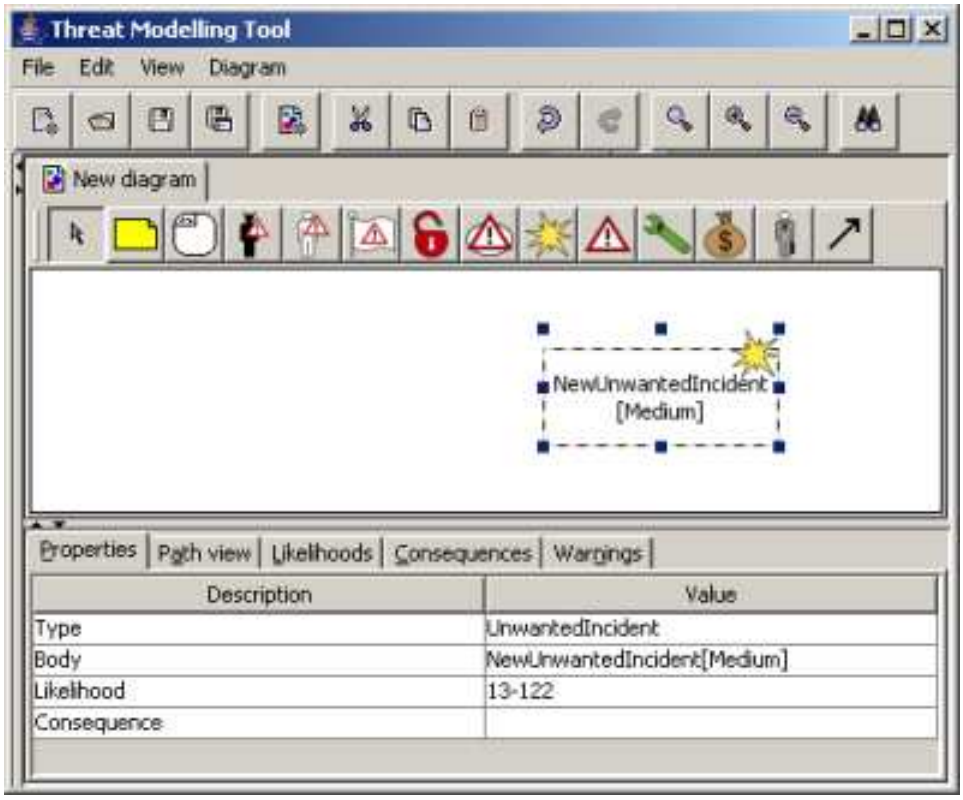

Figure 5.1: The SCORE tool interface

Furthermore, the tool helps the risk analysis leader to communicate with the risk analysis participants and to collaborate with the risk analysis secretary. The risk analysis participants will experience that it is easier to relate to the scenarios under evaluation since they only need to concentrate on the lower level threat scenarios instead of the composite unwanted incidents.

5.1.1 Purpose

The SCORE tool aims to provide a user interface in the CORAS tool for the risk analysts to simplify the process of risk estimation, meaning that the tool serves the purpose of:

- providing an interface for the analysts to document risk estimations based on expert judgements
- automating risk estimation calculations (on-the-fly) based on the SCORE method (chapter 4)

- providing an overview of the properties of the current element of interest
- providing warnings when inconsistencies with respect to risk estimations occur
- automatically importing predefined estimation values from the CORAS tool
- allowing the analysts to choose legal estimation values as input

In the next section we describe how the tool implements the defined method from chapter 4 to accommodate the requirements for automated risk estimation calculations.

5.2 Software requirements

Software requirements engineering is an important phase in a software development life cycle [30]. Normally in an industrial setting, a requirements document would act as a contract between the customer and the developer(s) where the delivery in detail is specified. In this report the requirements act as success criteria upon which the final results are evaluated (conducted in chapter 7). The requirements are nevertheless important for the system thus been developed as an own *Software Requirements Specification (SRS)* document in appendix B. Below we present the main concepts of this document, and where the SRS uses precise textual notion for implementation detail accuracy, we introduce use case diagrams and graphical models in addition for clear and simple perception.

5.2.1 User requirements

The user requirements concerns legal operations the users can execute. They are presented in terms of a use case diagram (figure 5.2) followed by a short textual description for each of them.

Users

The actor from figure 5.2 is defined as the user of the CORAS tool:

Risk analysis secretary - The user of the CORAS tool

A CORAS risk analysis entitles two roles (at least, there could preferably with available resources present be more) where one is to be the risk analysis secretary.

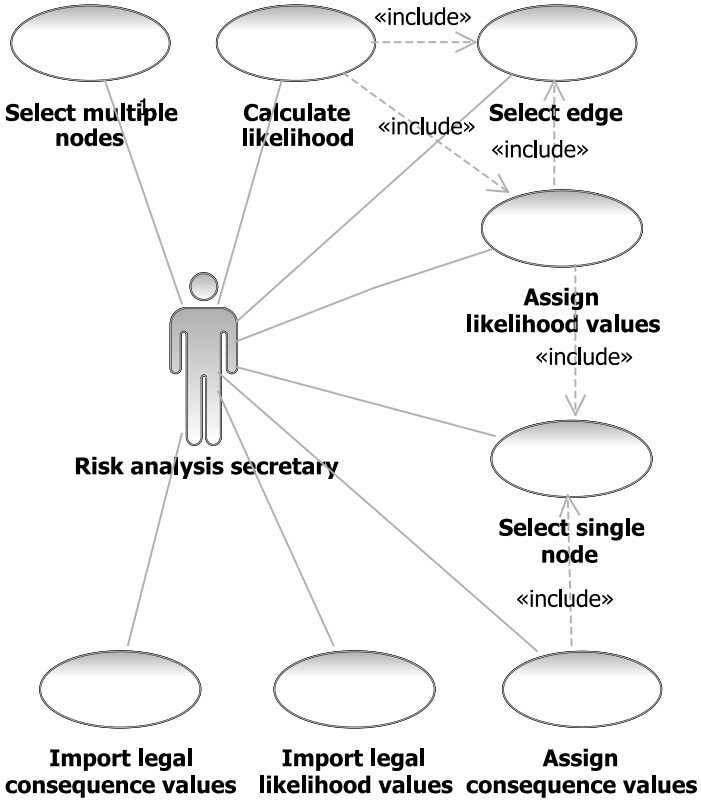

Figure 5.2: Use case diagram

The secretary's responsibility involves using the CORAS tool to document the risk analysis both live at the risk workshops and editing before and after.

Use cases

UC-1 - Select single node
UC-2 - Select edge
UC-3 - Select multiple diagram elements
UC-4 - Assign likelihood value
UC-5 - Assign consequence value
UC-6 - Calculate likelihood value
UC-7 - Import likelihood values
UC-8 - Import consequence values

The first three use cases (UC-1, UC-2 and UC-3) deal with the simple action of selecting or highlighting a diagram element (in other words: the element gains

focus/marquee selection). The response of the use cases is that the tool shows the appropriate properties for the selected element(s) with its respective features. These features are the assignment of likelihood or consequence values – use cases UC-4 and UC-5 that are made available through the same property view. UC-6, calculate likelihood value, is available once likelihood values have been assigned. Finally we have the import use cases UC-7 and UC-8 that allows the user to import predefined likelihood or consequence values from the CORAS tool.

5.2.2 System requirements

Given the required use cases as starting point we establish the system requirements through low-level detailed and textual specifications concerning the technical aspects of the system. The SRS (appendix B) provides a detailed examination of the system requirements, and below we present the summary of the most central requirements.

The SCORE tool shall:

SR-1 - provide a pane for the various risk estimation functions containing a *property view, path view, problem view, likelihood view* and a *consequence view*

SR-2 - display properties in the property view when a single node is selected The properties are:

- type – the element type
- body – content of the element
- likelihood – the element's assigned likelihood value
- consequence – the element's assigned consequence value

SR-3 - display properties in the property view when an edge is selected The properties are:

- type – the edge type
- likelihood – the edge's assigned likelihood value
- source – the initiating element
- target – the initiated element

SR-4 - display selected path in the path view when multiple nodes are selected

SR-5 - allow for likelihood values to be assigned to a node or edge when selected

SR-6 - allow for consequence values to be assigned to a harm edge when selected

SR-7 - automatically calculate likelihood on assignment or changes in likelihood values

System requirement 1 (SR-1) declares a panel in the CORAS tool that is the most conspicuous visual result of the SCORE tool. The pane consists of different tabs, and the two following requirements SR-2 and SR-3 describes functionality for one of them, the property view. SR-4 deals with the path view while the last three requirements describes features related to functionality. SR-5 and SR-6 considers assignment of likelihood and consequences respectively while SR-7 defines the core likelihood calculation of estimated values.

Chapter 6

The SCORE tool

One of the two main results from the SCORE project is the computerised tool that supports the method for risk estimation. The SCORE tool application will be available at [31] for a limited period. Here we present the implementation of this. When we talk about the computerised tool in this chapter, we use the notion *the SCORE tool* or simply *the tool*. Note that this should not be confused with the original *CORAS tool* which this tool is an implemented part of. We therefore use the full name when we refer to *the CORAS tool*.

The first part of this chapter introduces an overall picture of how the SCORE tool implements the SCORE method. Then we present the tool from a high-level user point of view with live screen shots that documents the final implementation. We go further into detail under the design section where we examine the architectural design through UML2.0 class diagrams and UML2.0 sequence diagrams. Finally we give a more technical presentation of the implementation specific details.

The first two sections of this chapter should be readable by anyone, but aims to address users familiar with the CORAS tool. When we come to the design section, software modelling knowledge is required and it is meant for readers with interests in the underlying architectural design, whereas the latter section is dedicated towards people with programming experience, primarily with J2SE or equivalent.

6.1 How the tool implements the SCORE method

Chapter 4 defines a set of predefined rules that describes the behaviour of the tool. Here we give a detailed account of how the tool supports the SCORE method driven by an example of the *1-to-1 initiate relationship* rule.

However, the *1-to-1 complete* rule 4.2.1 is not implemented solely, but is combined with the *1-to-1 incomplete* rule (4.2.3) since we in a real setting always account for incompleteness.

The rule yields:

$$n_1(l_1) \xrightarrow{p} n(l)$$
where
$$l \geq (l_1 \times p)$$

The main purpose of this rule is to fulfil the use case:

[UC-6]: Calculate likelihood value l of n [1]

where l is the output from the SCORE tool. In order to calculate l, the tool is dependent of that the user has assigned the appropriate likelihood values as input, namely the use cases:

[UC-5]: Assign likelihood value l_1
[UC-5]: Assign likelihood value p_1

which is the same use case performed both at the diagram element n_1 and the edge e between n_1 and n that holds the likelihood value p_1. Again, these are dependent of the distinct use cases, respectively:

[UV-1] Select single node n_1
[UV-1] Select edge e

6.2 Graphical user interface (GUI)

The visual outcome from the SCORE tool work is presented at a user-level point of view through screen shots of the tool assisted with comments and explanations.

[1] For the use case number, refer to section 5.2.1.

The first at glance and most obvious evidence of the SCORE tool, is the new panel appearing at the bottom of the diagram window of the editor (the total interface is presented in figure 5.1 under the tool vision section 5.1). When the tool starts up, the *Properties* tab is visible by default. The whole panel can be minimised to maximise the diagram window, and the user can choose between the tabs: *Properties*, *Path view*, *Likelihoods* and *Consequences*. They will all be presented in detail below.

Property view

Figure 6.1 shows the property view panel with the properties of an unwanted incident. The *Value* column on the right hand side consists of editable fields, except for the *Type* row that is fixed according to the diagram element type. The *body* row takes pure text as input, while the *Likelihood* and *Consequence* rows are drop-down boxes.

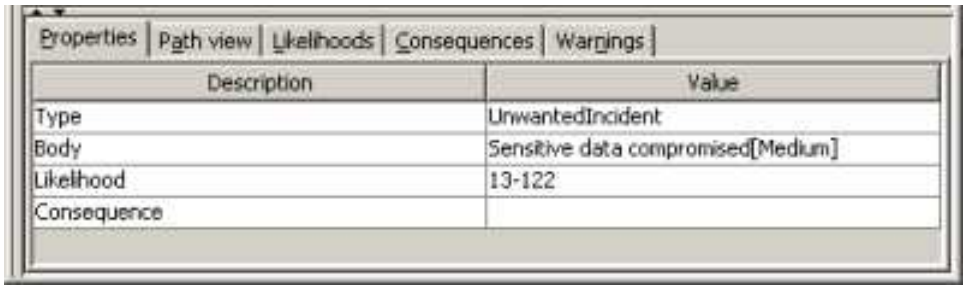

Figure 6.1: The property view panel

Assign likelihood

To assign a likelihood value to the selected unwanted incident, the user can use the drop-down menu in the property panel at the *Likelihood* row as shown in figure 6.2 below.

Figure 6.2: The likelihood values drop-down box

The drop-down menu only contains legal predefined likelihood values, and when selected the diagram element gets updated on-the-fly. The contained likelihood values in the drop-down box are defined in the likelihood view (below) and vary between numerical and linguistic representation according the user's choice.

Define likelihoods

Under the tab *Likelihoods* we find a table containing the likelihood definitions (figure 6.3). The leftmost column defines the linguistic expressions to which

Figure 6.3: Likelihood definitions

the corresponding other two numerical ones are mapped. The user may self define whether to express the likelihoods in a numerical or linguistic fashion by right-clicking the *Likelihoods* tab (figure 6.4). The *Likelihood* and *Probability* columns define a finite set of disjoint intervals that the user may self redefine.

Upon modification they become immediately visible in the drop-down box for likelihoods in the property view according to what diagram element is selected. When the user selects a relationship, the drop-down box contains the probability values, and when either unwanted incidents or threat scenarios are selected it contains frequencies (assumed that numerical view is chosen, otherwise the linguistic values applies to either or). Note that likelihoods are always stored as numerical expressions in the diagram elements. However, the linguistic value shown at the diagram element is only a transformed value from the numerical ones.

Figure 6.4: Choose between numerical and linguistic likelihood expression

Calculate likelihood values – 1-to-1 initiate relationship

If we assign likelihood values to both the threat scenario and the initiating edge in figure 6.5, we may choose to automatically calculate the likelihood value of the unwanted incident. The calculation is simply executing by right-clicking the unwanted incident and then choose the *Calculate likelihood* option. Upon successful calculation the result is automatically stored as the likelihood of the unwanted incident.

Calculate likelihood values – many-to-1 initiate relationship

The scenario from above may further be extended to include several threat scenarios (or other unwanted incidents for that matter). The calculation is executed the same way, but the results rely on a somewhat more complex calculation algorithm. Figure 6.6 proves the results of a many-to-1 initiate relationship calculation.

Remaining tabs

The *Consequence* tab contains a table similar to the *Likelihood* tab and their functions are analogues. Defined consequences are updated the same way and are

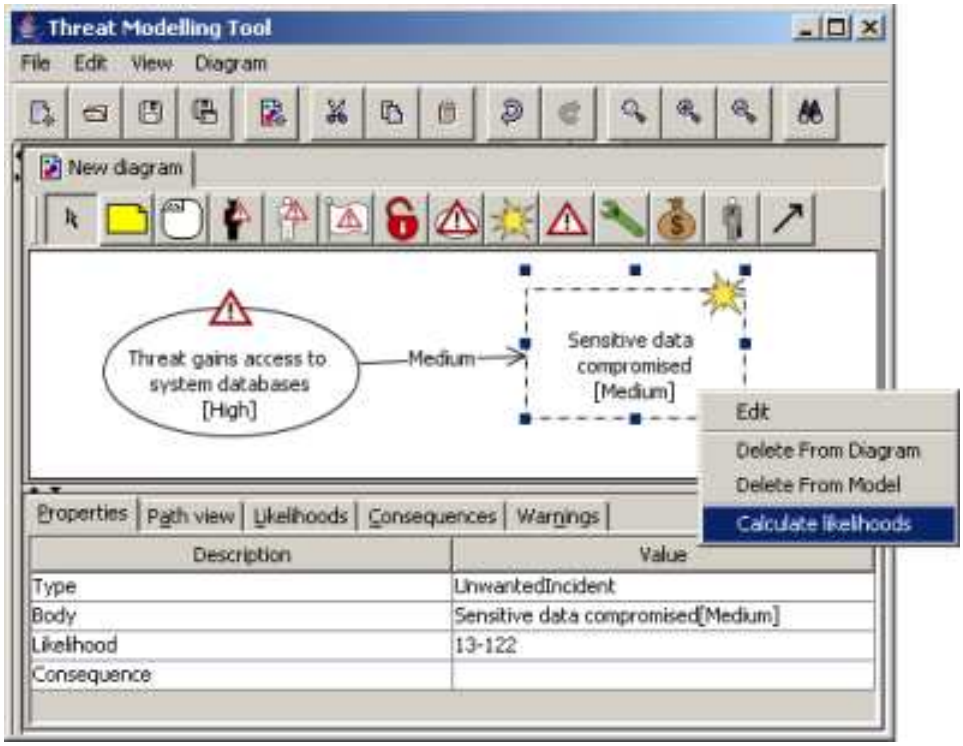

Figure 6.5: Aggregating likelihood value on a 1-to-1 initiate relationship

present in the drop-down box of the *Consequence* row in the *Properties* panel. The only difference is that the consequences naturally apply to harm relationships instead of initiate relationships.

Under the *Warnings* tab we find a simple listing of warnings produced by the system to the users. Warnings are created when the user defines illegal likelihood or consequence intervals (i.e. they contain negative numbers or are not disjoint) or when there exists inconsistencies in the diagram (i.e. two identical elements (by origin) contain different likelihood values).

The last tab *Path view* is a tab meant to display a selected path in the diagram. This feature is not implemented. However, the idea is that the user may select a number of elements contained in a path and assign likelihoods on these without being distracted by the rest of the diagram elements. When finished, the results of the current path's likelihood assignments and/or calculations are automatically inserted back into the rest of the diagram. Other affecting likelihoods will then be merged with these automatically.

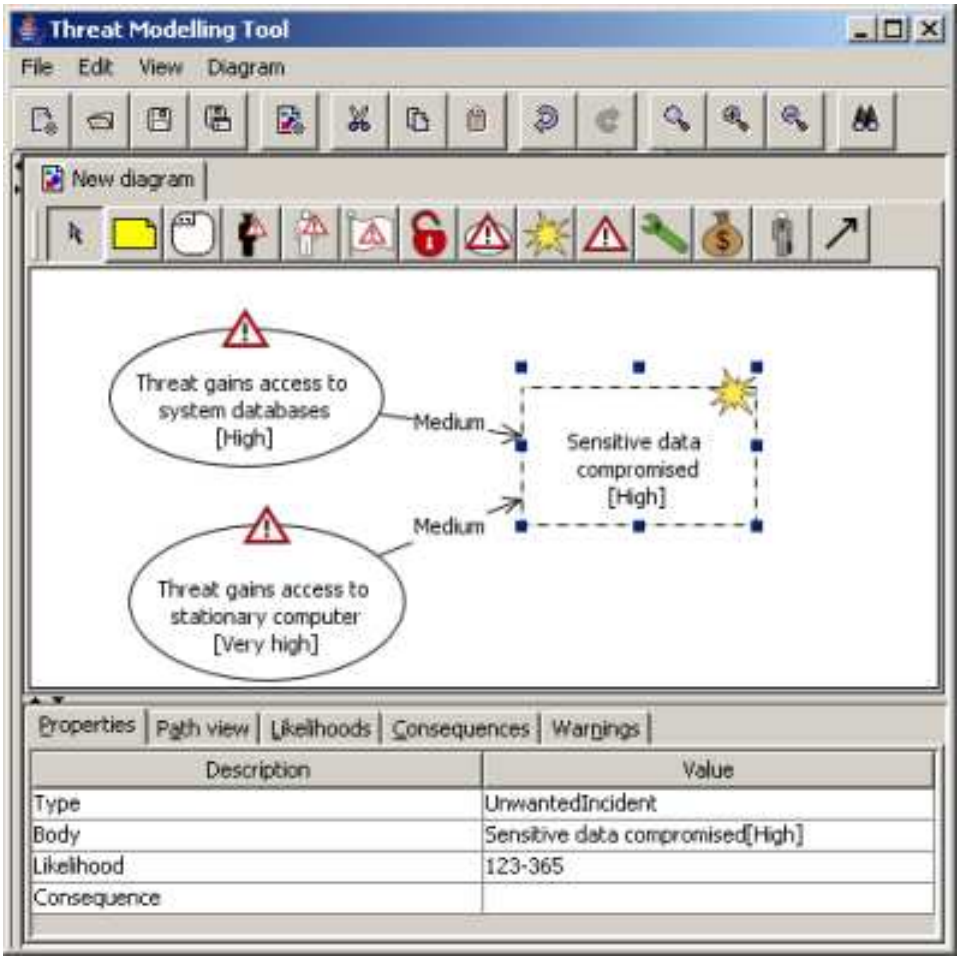

Figure 6.6: Aggregating likelihood value on a many-to-1 initiate relationship

6.3 Design

This section presents the internal architectural design of the SCORE tool as well as its dependencies to components in the CORAS tool, through different UML 2.0 modelling diagrams. First we give the context view with respect to the CORAS tool with an overall UML 2.0 component diagram and a brief presentation of the relevant features of the CORAS tool components. From section 5.2.1 we are familiar with the tool's use cases, and up next in this section we use UML 2.0 sequence diagrams to further specify two of the most central use cases. The remaining specifications may be found in appendix A. To support the sequence diagrams, we have a set of UML 2.0 composite structure diagrams that defines the internal structure of the system in a more general manner and with focus on the interconnected elements and their collaborations.

6.3.1 The SCORE tool and the CORAS tool joint component view

The existing implementation of the CORAS tool consists of several components working together in a complex manner. SCORE impacts only a few, but rather important components. Most important is of course, the *diagram-editor* under which the SCORE mainly operates. Figure 6.7 gives an overall overview of the collaborating components at the level of the diagram-editor.

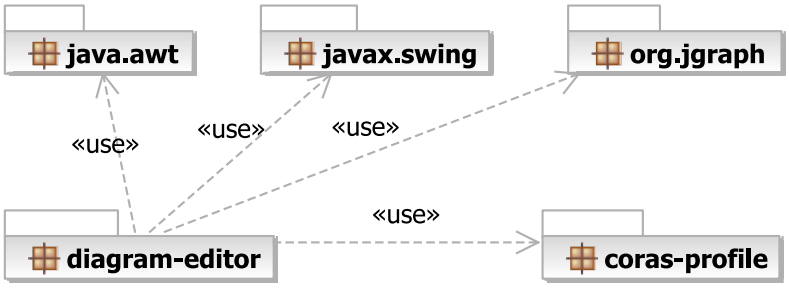

Figure 6.7: CORAS overview component diagram

The diagram-editor component

The diagram-editor is a stand-alone component that may be run completely without the rest of the tool. Its responsibility is to create or edit CORAS diagrams and store them in a user defined `.dgx` file or import them into the tool. The SCORE tool is implemented completely under the diagram-editor package. It uses the coras-profile to update CORAS diagram elements the same way as the rest of the diagram-editor.

The coras-profile component

The coras-profile is the package that stores all information about the various diagram elements. All elements are organised in a hierarchical tree structure originating from a common `coras.profile.diagram.DiagramElement` and further divided into nodes and edges. The diagram-editor represents its diagram elements in a `org.jgraph.graph.GraphCell` type that is mapped against the leaf nodes of the coras-profile. The nodes of whom we concern are the `UnwantedIncidentNode`, `ThreatScenarioNode`, `InitiateEdge` and `Harmedge`.

6.3.2 The SCORE tool architectural design

To understand the internal structure of the SCORE tool, we should first have a look at how the diagram-editor is constructed and where the SCORE tool occurs. Figure 6.7 outlines the true structure of the CORAS tool as it is implemented originally with actual java packages and their relations. In order to not getting distracted by implementation details at the moment, we need to illustrate the architectural design of the system. Therefore we choose to use composite structure diagrams that allow us to design the structural design with respect to the system elements and their relations without being implementation specific. Implicitly, this means introducing composite structures that exist within a dedicated class as to what is defined for a composite structure diagram, even though they do not exist in the implementation.[2]

Composite structure - level 1: CORAS main

Below we model the component diagram from figure 6.7 above using a composite structure diagram. The composite structures are the components from before, but they have added ports that indicate communication functionality. From now we also remove all external dependencies such as the enterprise java and jgraph libraries to focus on the core business of the tool(s).

Composite structure - level 2: diagram-editor

The composite structure diagram in figure 6.8 outlines how the main components of the diagram-editor including the SCORE tool, are organised.

The composites `ProfileImpl`, `UI` and Diagram are the original composites from the diagram-editor. Their internal structure is mostly kept, but some modifications are of course required in order to embed the SCORE tool.

Composite structure - level 3: SCORE

If we examine the SCORE tool composite (figure 6.10), namely the `SCORE`, we see that the organisation from the original diagram-editor is kept. In addition, we have added a designated controller, the `RiskestimationController` to handle all communication between the `ProfileImpl`, `UI` and Method composites.

[2]However, when we get to the UML 2.0 class diagrams later, all classes are strictly identical to what is implemented, except the fact that they may be simplified due to readability.

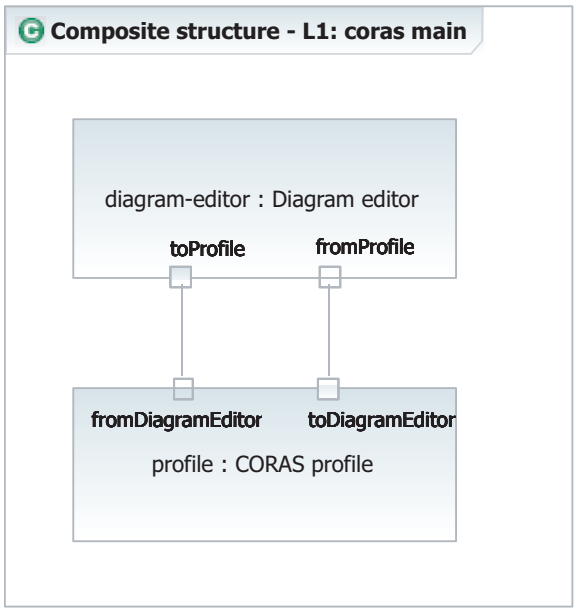

Figure 6.8: Level 1 composite structure diagram

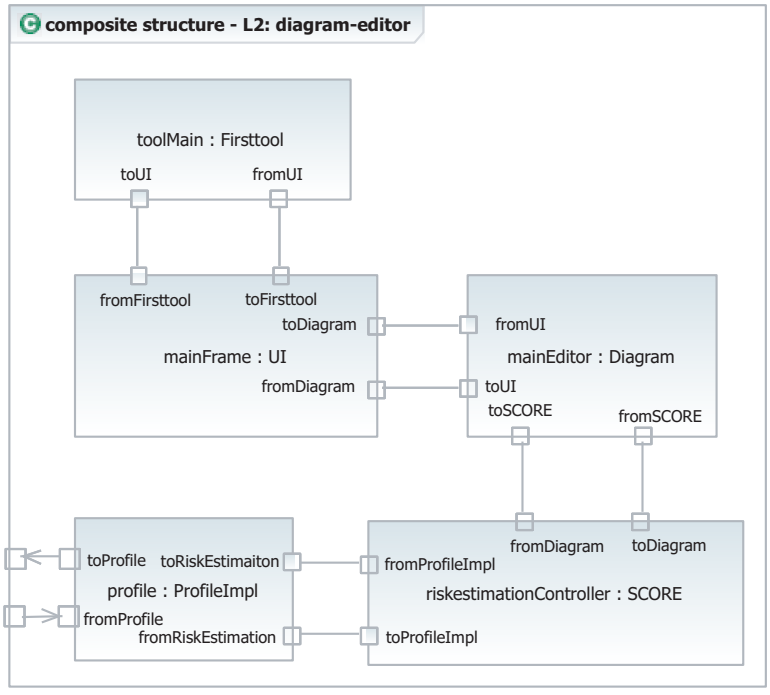

Figure 6.9: Level 2 composite structure diagram

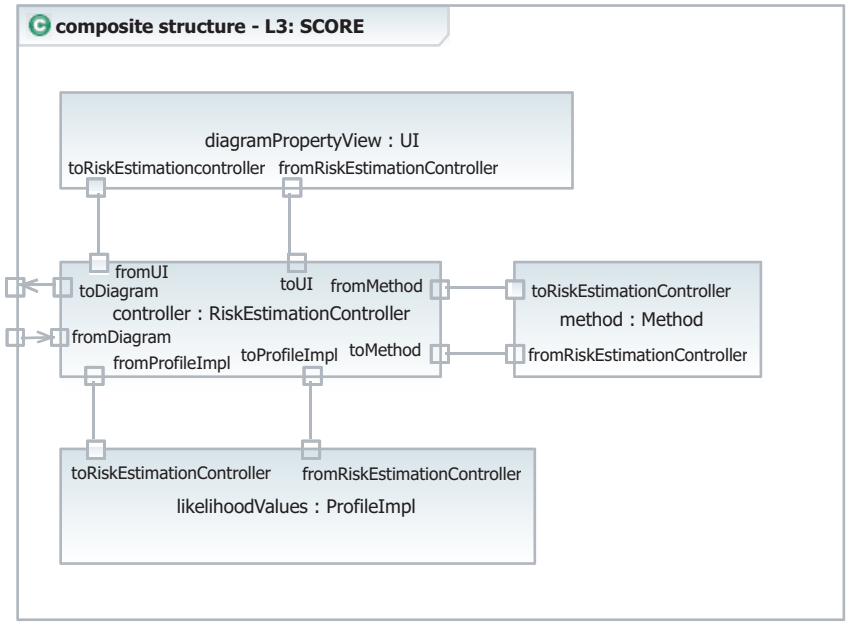

Figure 6.10: Level 3 composite structure diagram

In words, the controller receives user input from the `diagram-editor`and takes action according to what kind of user input. If the input requests a calculation, the controller forwards the data to the `Method` that does the actual calculations from which in turn is returned to the `diagram-editor` through the controller. Before the results are sent back to the `diagram-editor`, the controller makes sure to update the affected diagram elements under control by the `Diagram`.

The next level, level 4, provides the most detailed level of our implementation. We investigate the `UI` composite below, but the remaining composites only contain a few components, hence there is no need to display this graphically. The `RiskEstimationController` contains only a controller component, the `ProfileImpl` contains a `LikelihoodValues` and a `ConsequenceValues` component to store the respective values and the `Method` contains a component `LikelihoodCalculator` that processes the likelihood calculations and translations.

6.3.3 The SCORE tool behavioural design

We present the behavioural design of the tool using UML 2.0 sequence diagrams based on the composite structure shown above and the use cases from section 5.2.1. The use case *Assign likelihood value* is described to illustrate the most

important system processes. The presentation order follows the hierarchical levels from the composite structures.

Interaction diagram: Calculate likelihood value – level 0: user perspective

The risk analysis secretary interacts with the CORAS tool by choosing the feature *Calculate likelihood value* l_1. As response, he gets to see the diagram element's updated properties including the calculated likelihood value.

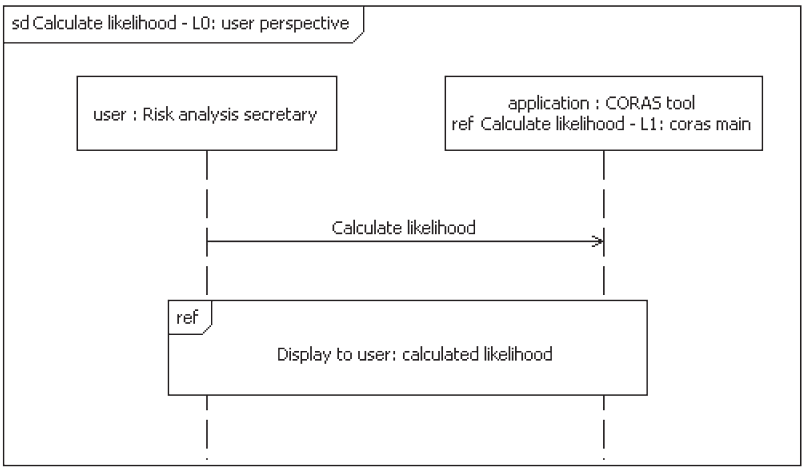

Figure 6.11: Level 0 interaction diagram - calculate likelihood value

Interaction diagram: Calculate likelihood value – level 1: CORAS main

The diagram-editor reacts to the request by getting hold of the CORAS profile element, processing the calculation and setting the new likelihood value.

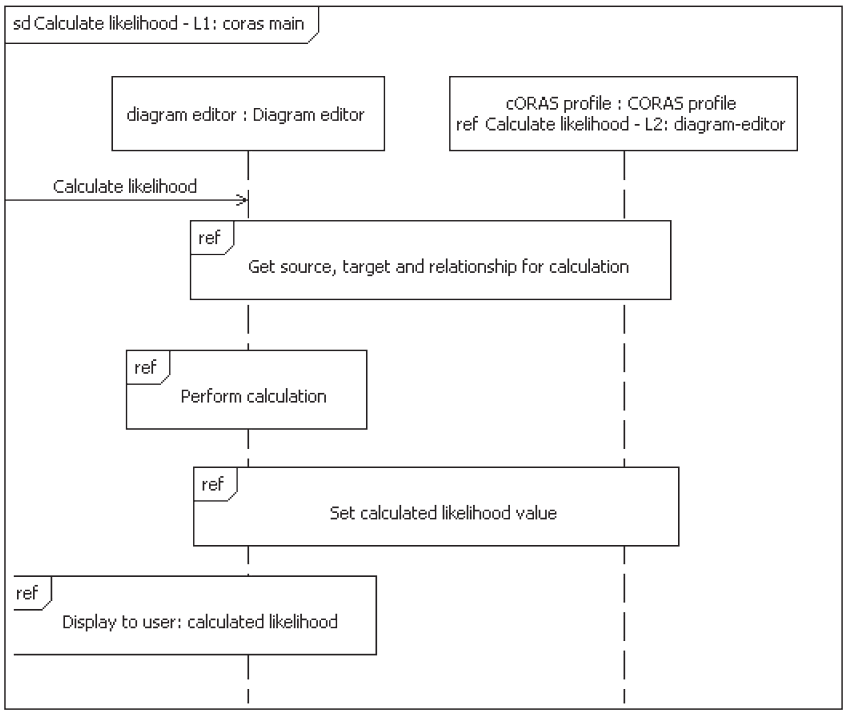

Figure 6.12: Level 1 sequence diagram - assign likelihood value

Interaction diagram: Calculate likelihood value – level 2: diagram-editor

Internally in the diagram-editor, the SCORE gets noticed of the likelihood calculation request. The controller gets the hold of the dependent graph cells and performs sufficient computations before setting the likelihood of the CORAS element and sees to that appropriate feedback is given to the user.

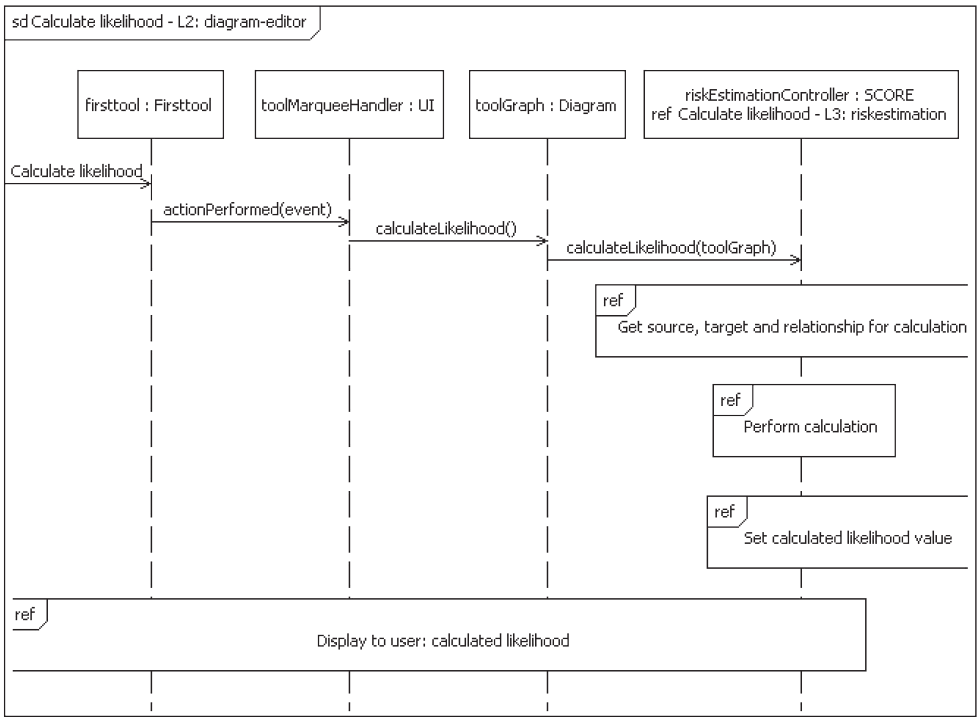

Figure 6.13: Level 2 interaction diagram - calculate likelihood value

Interaction diagram: Calculate likelihood value – level 3: risk estimation

The `RiskestimationController` handles all incoming likelihood change and processing requests. The first task is to get hold of the dependent source, and target or edge of the current diagram element as may be. The the controller sends the information to the likelihood calculator and gets a new likelihood value as response.

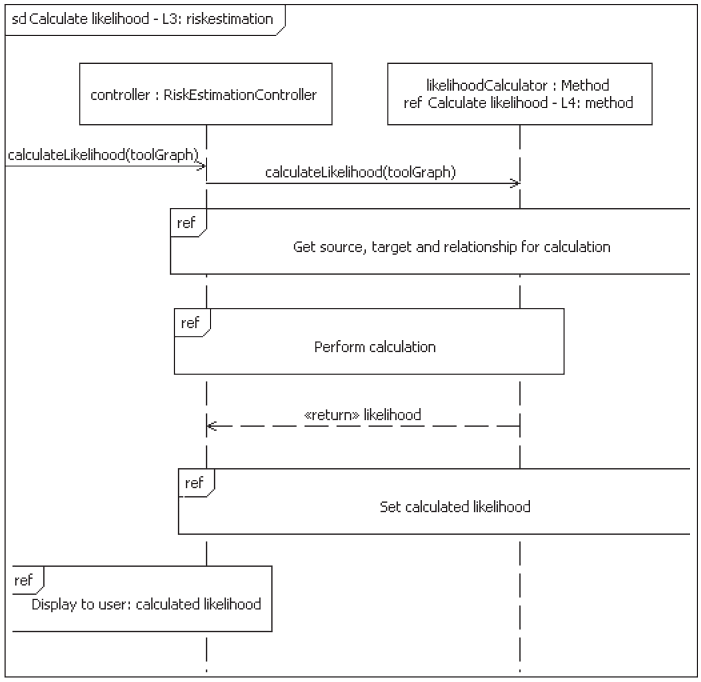

Figure 6.14: Level 3 interaction diagram - calculate likelihood value

Interaction diagram: Calculate likelihood value – level 4: likelihood calculator

A likelihood calculator receives likelihood values to be calculated, and this is where the SCORE method rules are applied. The calculator selects which rule to apply based on the input and returns the calculated likelihood value.

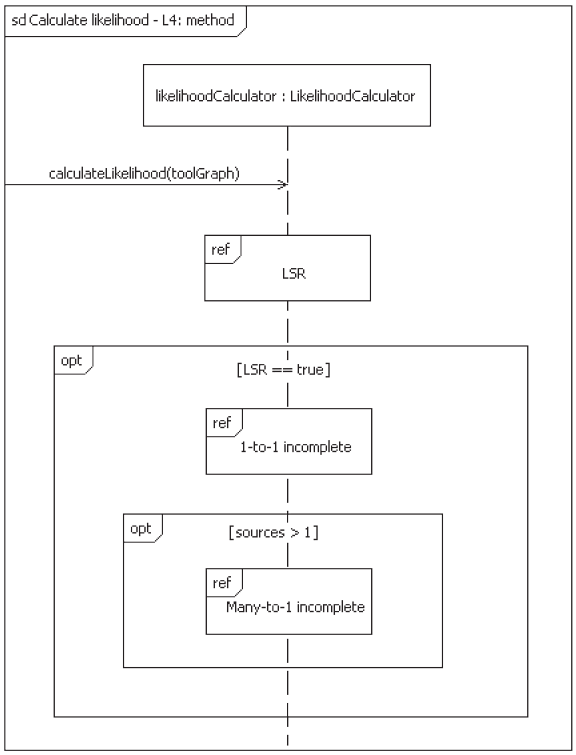

Figure 6.15: Level 4 interaction diagram - calculate likelihood value

6.4 Implementation

The SCORE tool is implemented using Java™2 Platform Standard Edition 5.0 (j2se5.0). As with the rest of the CORAS tool, the implementation is built and compiled with Maven 2.0 and developed using Eclipse SDK 3.2. The SCORE tool implementation relies heavily on the java packages `javax.swing` and `java.awt`, and `JGraph5.8` for graphical user interface as for the rest of the diagram-editor.

To present the implementation, we use a UML 2.0 class diagram in which we examine some of the most central classes. CORAS is an open source project, thus the full source code of the SCORE tool can be obtained through anonymous access to the CORAS SourceForge.net Subversion repository [32] , branched under `coras/branches/stig/src/modules/diagram-editor`. However this downloads the full `diagram-editor` source code and what is related to the SCORE work may be hard to isolate.

The overall class diagram

Figure 6.16 presents the overall class diagram for the SCORE tool. From the design presentation we recognise first of all the `RiskEstimationController` as the main controller class. Its main communication lines goes to the `DiagramPropertyView`, which is the interface towards the GUI of the `diagram-editor`, and the `LikelihoodCalculator` that obviously performs the calculations. Moreover the figure explains how the different tabs from the property panel are built, and what relations exists to the saved likelihoods and consequences.

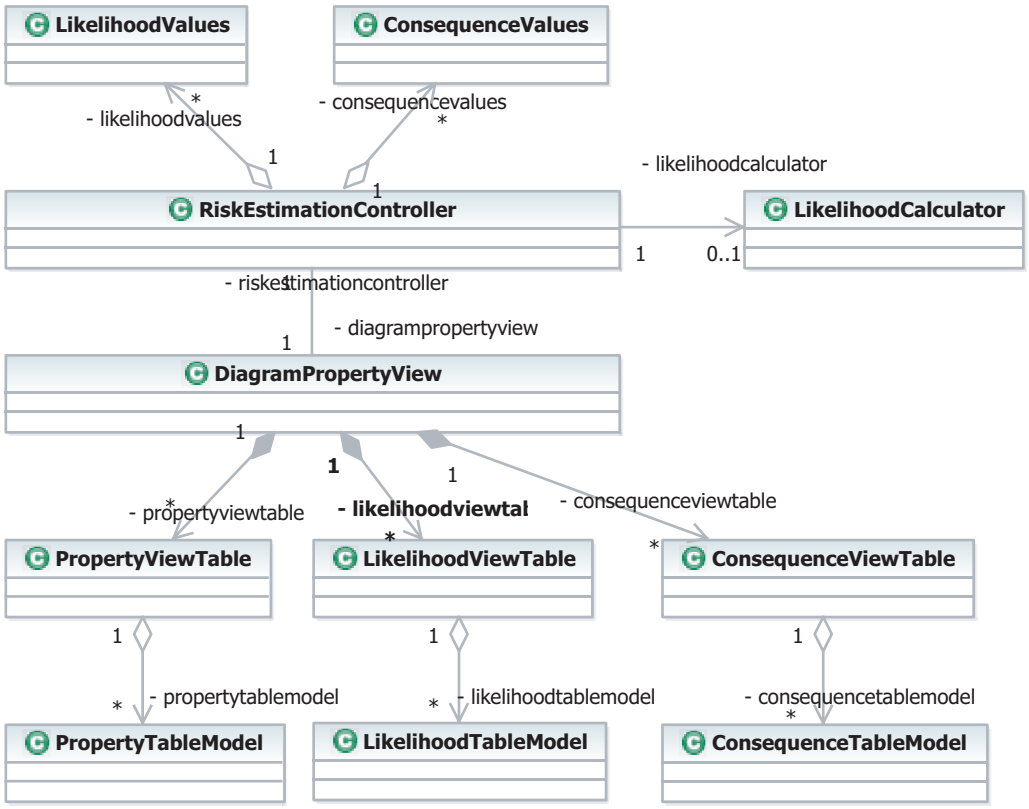

Figure 6.16: the SCORE tool overview class diagram

Changes in coras-profile

The diagram elements *unwanted incident*, *threat scenario*, *initiate relationship* and *harm relationship* need all to be assigned a likelihood value. The first two mentioned are sub classes of the `CorasGraphCell` and we easily add the private

attribute `likelihood` as a `String` representing the assigned likelihood. The two relationships may be assigned a likelihood the same way.

Asserted for all of the above is the format the likelihoods are stored in, namely a `String` representation of a numerical set. Even though the application user assigns linguistic likelihoods, these are translated into the numerical definitions in order to possibly perform calculations thereupon. When the diagram elements need to show the likelihood in a qualitative expression again, the numerical interval is mapped back to its corresponding linguistic value using the mean of the upper and lower bounds according to the SCORE method.

Class attributes and operations

The class diagram overview from figure 6.16 only explains the relations between the classes contained in the system. For more details, please refer to appendix A where selected classes are fully displayed with their respective attributes and operations.

Chapter 7

Evaluation

In chapter 3 we define a set of success criteria for SCORE. They provide the basis for the evaluation of the SCORE project in total. We also make use of the developed requirements specifications document B for the evaluation, but since this is developed with the purpose of specifying a fully functional system, they are not evaluated strictly.

In order to test to what extent the SCORE project fulfils its success criteria, we apply SCORE to the following evaluation strategies. First we present a case study that serves the purpose of the experimental simulation. The case is an industrial security risk analysis that applied the SCORE tool-supported method and was conducted with particular emphasis to this evaluation. Next we put SCORE into a more fixed laboratory setting where we try to provoke unexpected behaviour, before we apply a non-empirical evidence strategy that provides a walk-through of the tool-supported method.

7.1 Case study

This case study conducts a security risk analysis using the CORAS method assisted by the SCORE tool-supported method. The focus is in particular on the risk estimation process. The complete results of the risk analysis is a report that can be investigated in appendix C. Here we provide a brief summary and reproduce the most important findings from the risk analysis.

One presumption has been made before conducting the security risk analysis

7.1.1 Security risk analysis of the IT-system at Stubrud sjakk shop

The risk analysis is conducted in accordance with the CORAS method for Model-based Risk Analysis developed in the CORAS and SECURIS projects. The CORAS risk management process consists of the following five sub-processes:

Context identification – identify the context of the analysis. Describe the application(s) and/or organisation; identify usage scenarios, the assets of the target of analysis, and the risk evaluation criteria.

Risk identification – identify the potential threats to assets and the vulnerabilities of these assets. Identify unwanted incidents.

Risk estimation – estimate the consequence and frequency value of each unwanted incident identified in sub-process 2.

Risk evaluation – identify the level of risk associated with the unwanted incidents already identified and assessed in the previous sub-processes.

Risk treatment – address the treatment of the identified risks, and how to prevent the unacceptable risks.

Each of the phases will be examined more or less below. We emphasise the risk estimation phase, but the remaining will only be summarised sufficiently enough for the reader to understand the basics and the risk estimation phase.

Context identification

The risk analysis is performed at the e-commerce web-shop *Stubrub sjakk shop* (sjakk = chess), from now simply referred to as the *sjakk shop*. The company is a sole proprietorship and is marketed mainly through its website corporate `www.sjakk-shop.no` where it makes most of its sales turnover. The manager (and sole proprietor) runs his business mostly on his own from his home office. This is also where all physical assets like product stock and of course office facilities are located. Although the web hosting of the online website and the accounting system is outsourced to a third party, the office holds business critical tools such as a local computer system. Figure 7.1 shows the relevant elements that constitutes the environment around the target of evaluation (TOE).

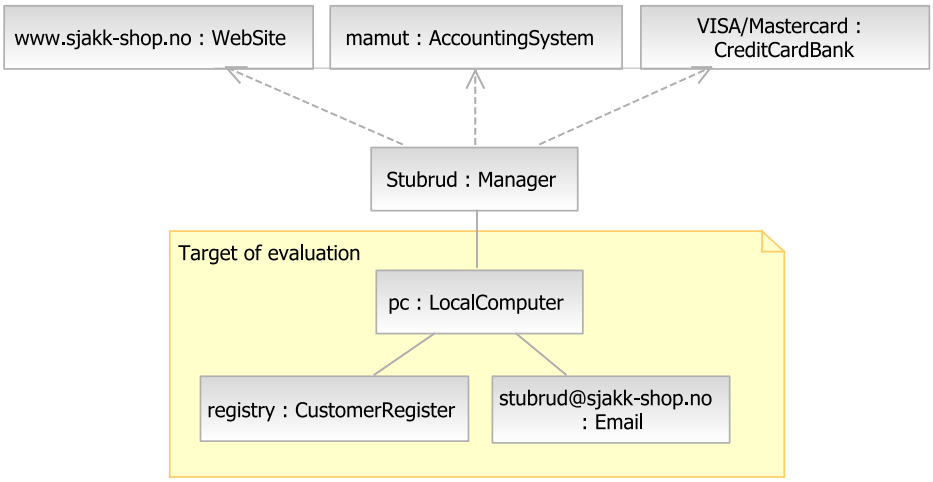

Figure 7.1: Target of evaluation and environment

Target of evaluation concerns the local computer system that the manager uses to carry out the everyday duties of the sjakk-shop. We have identified two assets of the Sjakk shop that the risk analysis is conducted with respect to:

Asset ID	**Description**	**Value**
E-mail/customer register confidentiality	The confidentiality of the company's e-mail and customer registry located at the local computer system	375 000,-
E-mail availability	The availability of the sjakk-shop e-mail	25 000,-

Table 7.1: Asset table

The context identification phase involves defining likelihood and consequence values and the risk evaluation criteria. Note the probability values (table 7.3) that play an important part in the field trial due to the significance for the SCORE project (their influence will be explained and discussed later). The consequence values are given in table 7.4 and 7.5 respective for each of the assets.

Frequency values (Normalised values pr. year in parenthesis)	
Almost never	Yearly (< 1)
Very seldom	Monthly (1 – 12)
Seldom	Weekly (13 – 122)
Often	Every 1-3 days (123 – 365)
Very often	Daily (> 365)

Table 7.2: Frequency values

Probability values Distribution domain in range [0 – 1]	
Very low	< 0 – 0.1]
Low	< 0.1 – 0.3]
Medium	< 0.3 – 0.6]
High	< 0.6 – 0.8]
Very high	< 0.8 – 1]

Table 7.3: Probability values

Consequence values Loss of income (1000 NOK) (E-mail/customer register confidentiality)	
Insignificant	0 – 10
Moderate	11 – 25
Large	26 – 50
Critical	51 – 150
Catastrophic	> 151

Table 7.4: Consequence values:
E-mail/customer register confidentiality

Consequence values Inaccessible time (E-mail availability)	
Insignificant	0 to 3 hours
Moderate	4 hours to 1 day
Large	2 to 5 days
Critical	6 to 14 days
Catastrophic	15 days +

Table 7.5: Consequence values: E-mail availability

In the risk evaluation phase the risk evaluation criteria are used in practise (figure 7.6 and 7.7), but we leave them out here to save space. The sjakk-shop is a company with high focus on security and does not accept a high level of risk, especially with respect to the *e-mail/customer register confidentiality* asset. Figure 7.6 proves this point by not accepting more than moderate risks at medium frequency. Likewise with the asset *e-mail availability* (figure 7.7), but since this one has lower priority, we accept a slightly higher risk level.

Risk identification

The risk identification phase has for its object to reveal and identify possible risks to the predefined assets. This is done by first to define what unwanted incidents could occur in our context. Next we deduce their causes by identifying existing and/or possible threats and vulnerabilities. The results of this phase, are the diagrams under the risk evaluation phase below (figure 7.2 and 7.3), except without the likelihood and consequence values.

Risk evaluation

The identified risks from the previous phase are now evaluated with respect to the probability for the risks to occur and the possible impact they would have to the respective asset. In practise, we assign the unwanted incident with likelihood and consequence values, and the risks can now be evaluated with respect to the risk evaluation criteria we defined in the context identification

7.1.1. The diagram below shows the result of this phase, and they are the same threat diagrams as the previous phase defined, but with the mentioned values assigned.

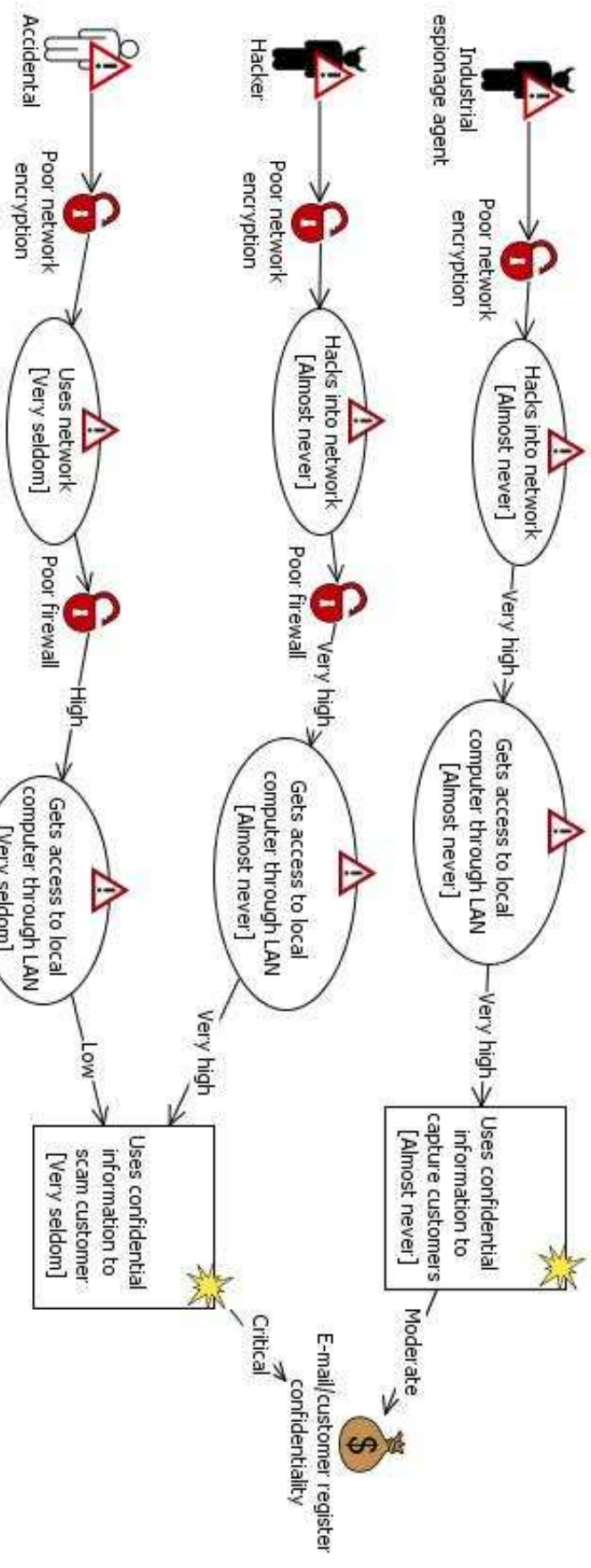

Figure 7.2: Customer registry confidentiality

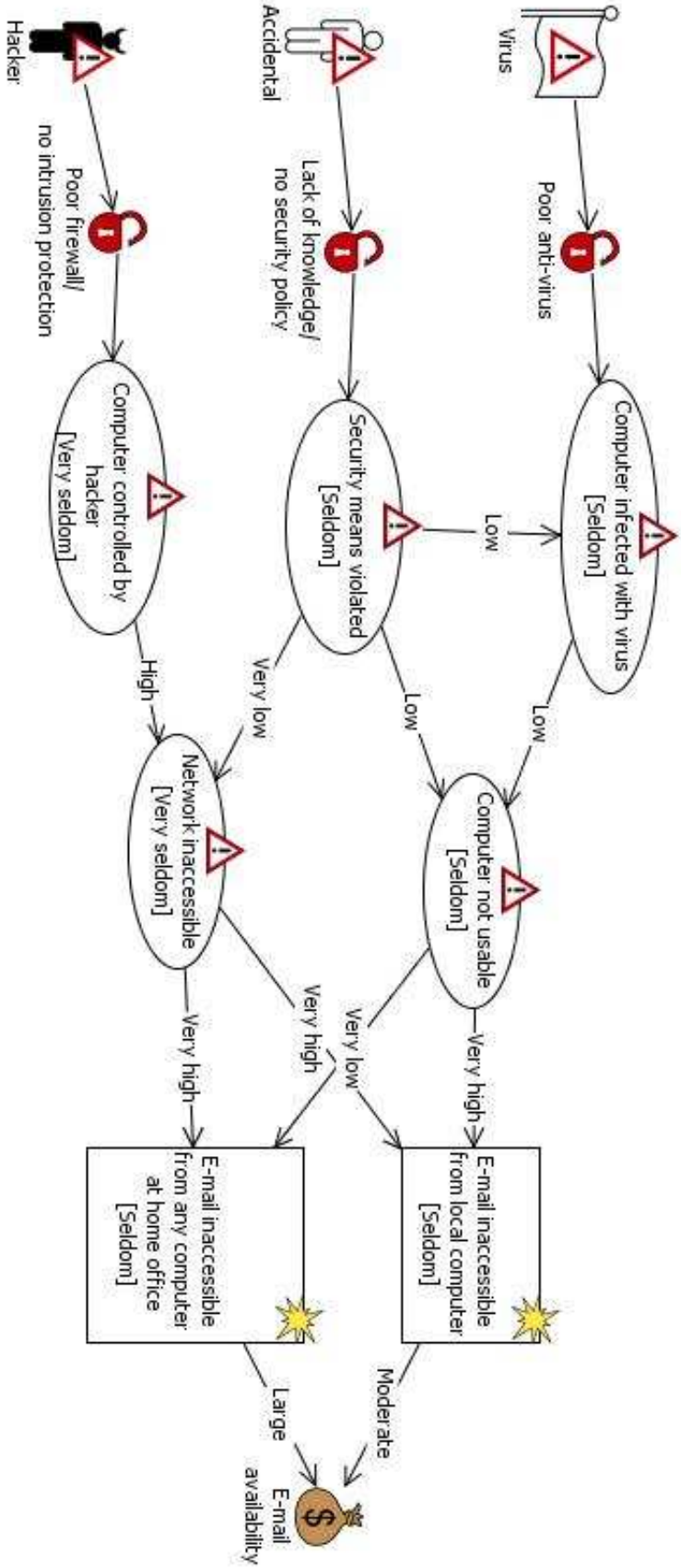

Figure 7.3: E-mail availability

In order to be able to rate the risk, we name them as follows:

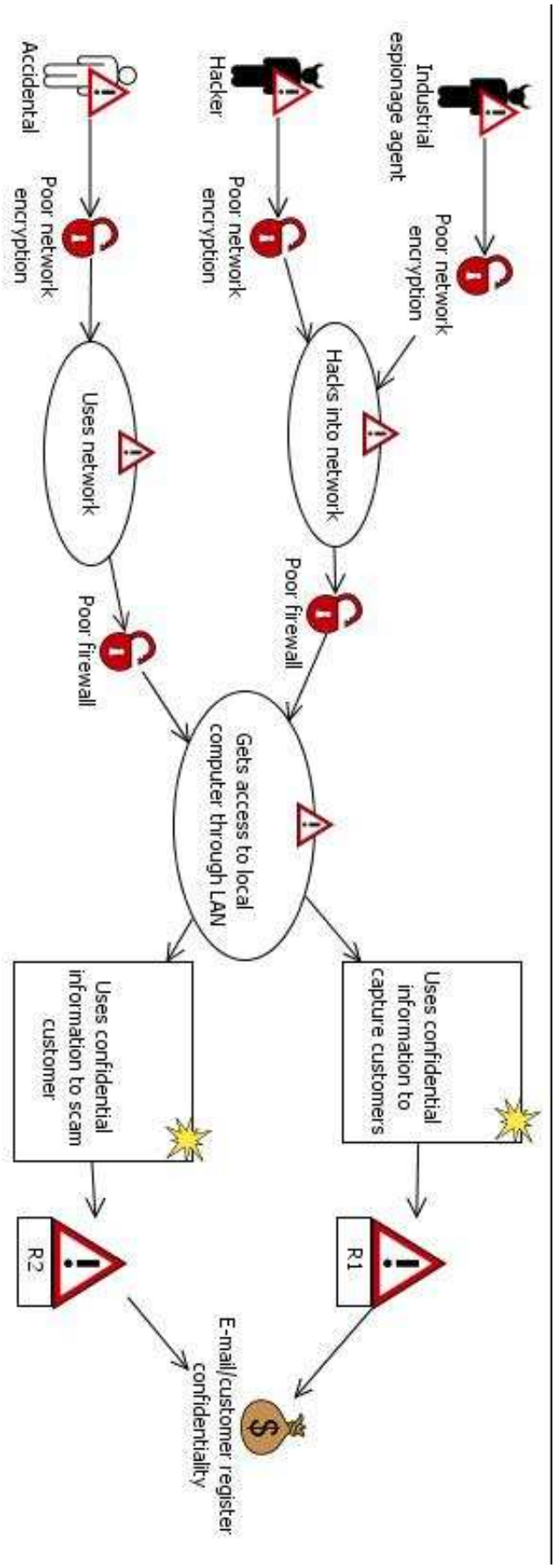

Figure 7.4: Customer registry confidentiality

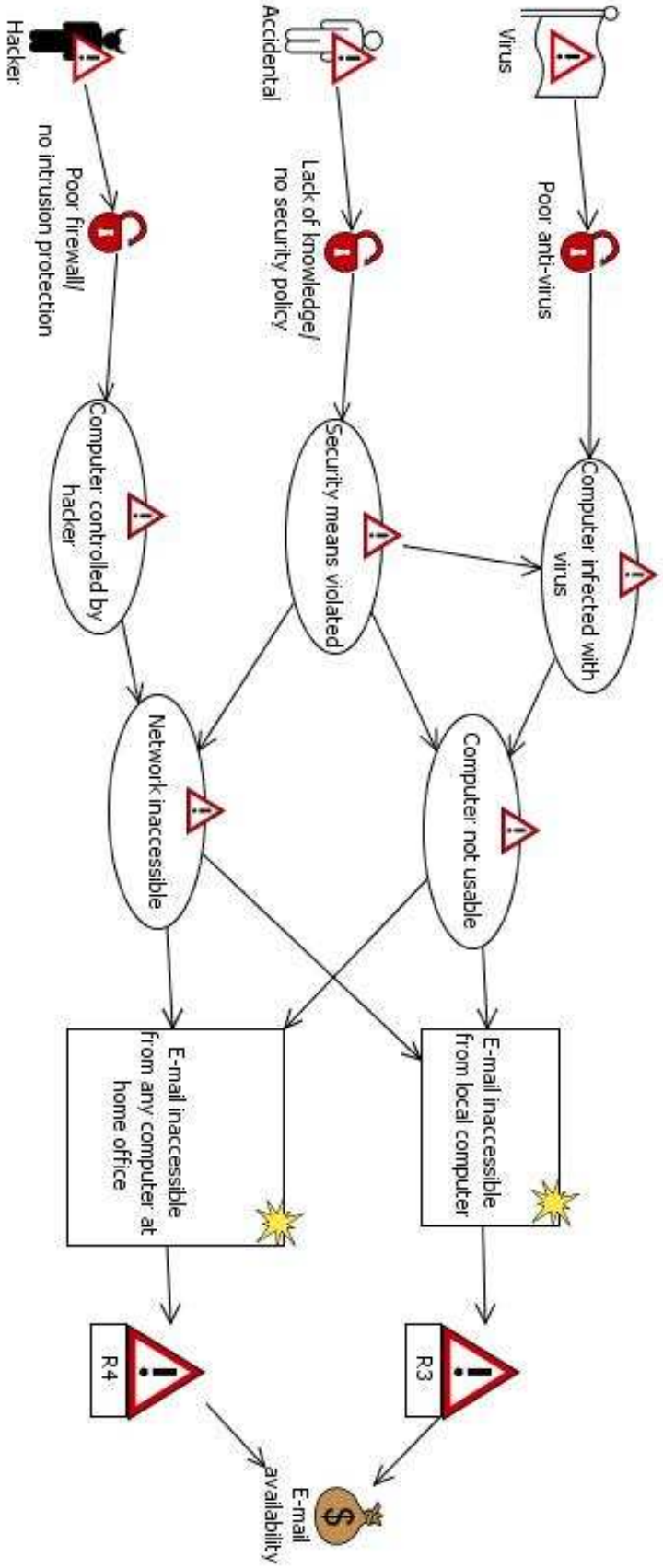

Figure 7.5: E-mail availability

When the risks are inserted into the risk evaluation criteria matrices, it is easy to see which risks are subject to treatment evaluation, namely the risks **R2**, **R3** and **R4** that fall within the red or dark shaded areas:

Loss of income (E-mail/customer register confidentiality)					
	Frequency				
Consequence	Almost never	Very seldom	Seldom	Often	Very often
Insignificant					
Moderate	**R1**				
Large					
Critical	**R2**				
Catastrophic					

Figure 7.6: Risk evaluation criteria matrix: Loss of income

Inaccessible time (E-mail availability)					
	Frequency				
Consequence	Almost never	Very seldom	Seldom	Often	Very often
Insignificant					
Moderate			**R3**		
Large					
Critical			**R4**		
Catastrophic					

Figure 7.7: Risk evaluation criteria matrix: Inaccessible time

Treatment

Refer to appendix C for the treatment results.

7.1.2 Case study experiences

During the risk analysis the risk analysis participants made a number of experiences summarised below.

Feedback from field expert

The field expert (Stubrud) had the following responsibilities during the risk analysis (amongst others – we emphasise tasks related to the risk estimation phase):

- provide the context identification including definition of frequency, likelihood and consequence scales
- identify the threat picture by creating threat diagrams
- assign likelihood values to the identified threat scenarios (note: not all of them, but the ones identified as leaf nodes of a fault tree)
- assign probability values to **all** of the initiate relationships
- assign consequence values to the harm relationships

After the risk analysis was finalised, the field expert (Stubrud) was asked to review the results related to the risk estimation and compare them to his expectations. He was asked the following questions:

> *With respect to each of the respective risks (R1,R2, R3 and R4); how do the general results from the risk estimation phase correspond to your expectations? Do their locations in the risk matrices seem reasonable, or would you expect a different result?*

> *With respect to each of the identified unwanted incidents; do the assigned likelihood values seem reasonable based on the input you provided for the threat scenarios, or do any of them seem to have a too high or too low likelihood?*

His response can be summarised in a generic manner:

> *The general results of the risk estimation seem reasonable. R1 would be OK to accept, hence it is appropriately located at the accepted area in the risk evaluation criteria matrix (figure 7.6). R2 in the same matrix needs evaluation, but is located only one step from the accepted area, which seems*

within reason. The second risk evaluation matrix (figure 7.7) however, may be questioned. Firstly the risk R3 located one step from the accepted area and subject to evaluation is approved and OK. The last risk, R4, may be evaluated somewhat too high in frequency. It is more likely to believe that it should be located only one step from the accepted area than two steps that is the current situation.

Feedback from risk analysis leader

The risk analysis leader (Torsbakken) had the following responsibilities during the risk analysis (amongst others – we emphasise tasks related to the risk estimation phase):

- document the context identification into the CORAS tool including definition of frequency, likelihood and consequence scales
- document and support the threat picture identification through creating threat diagrams using the CORAS tool
- document the likelihood values assigned to the threat scenarios
- document the probability values assigned to the initiate relationships and making sure all of them were covered
- document the consequence values assigned to the harm relationships

Due to the limited available resources for the risk analysis, the risk analysis leader also held the title of the risk analysis secretary thus additionally performing her responsibilities (i.e. documenting the risk analysis results).

Response from the security analysis leader with respect to the generated results of the risk estimation phase:

The results from the risk estimation phase seem fairly reasonable. The sjakkshop emphasises information security to a large extent and does not allow for a high level of risk acceptance. This comes obvious from the risk evaluation criteria matrices, but still the company sees its own improvement potential through the risks that are evaluated for treatment.

Response regarding the practical conduction of the risk estimation phase:

Since there existed practically no statistics or historical data regarding the identified unwanted incidents, the likelihood values needed to be assigned to the very first initiating threat scenarios all along. However, this was also the

purpose with respect to evaluation of SCORE. Compared to other analyses, it was a relief not having to worry about the composite unwanted incidents, but instead be able to target the initiating threat scenarios directly. Likelihood estimations seemed more accurate and reliable for the threat scenarios than they would have been for unwanted incidents.

7.2 Evaluation of the SCORE method

With the case study accounted for above kept fresh in mind we perform an evaluation of the SCORE method. We investigate the method rules successively and their underlying theories and exemplify this using the case study diagrams to illustrate the rules' application and presence.

In order to understand the results of the SCORE method applied to the Sjakkshop risk analysis, we need to further investigate the threat diagrams evolved from the risk evaluation. In the risk analysis itself (and also in its summary in section 7.1) we only display the linguistic risk estimation values, but in the following we illustrate what happens behind the scenes where the calculations from the SCORE method rules take place.

Figure 7.8 and 7.9 provides the complete overview of the diagrams with all likelihoods assigned, but here transformed into numerical values for the purpose of the evaluation. Subsequently we focus on the SCORE method rules one by one and explaining them with figure extracts from the two figures.

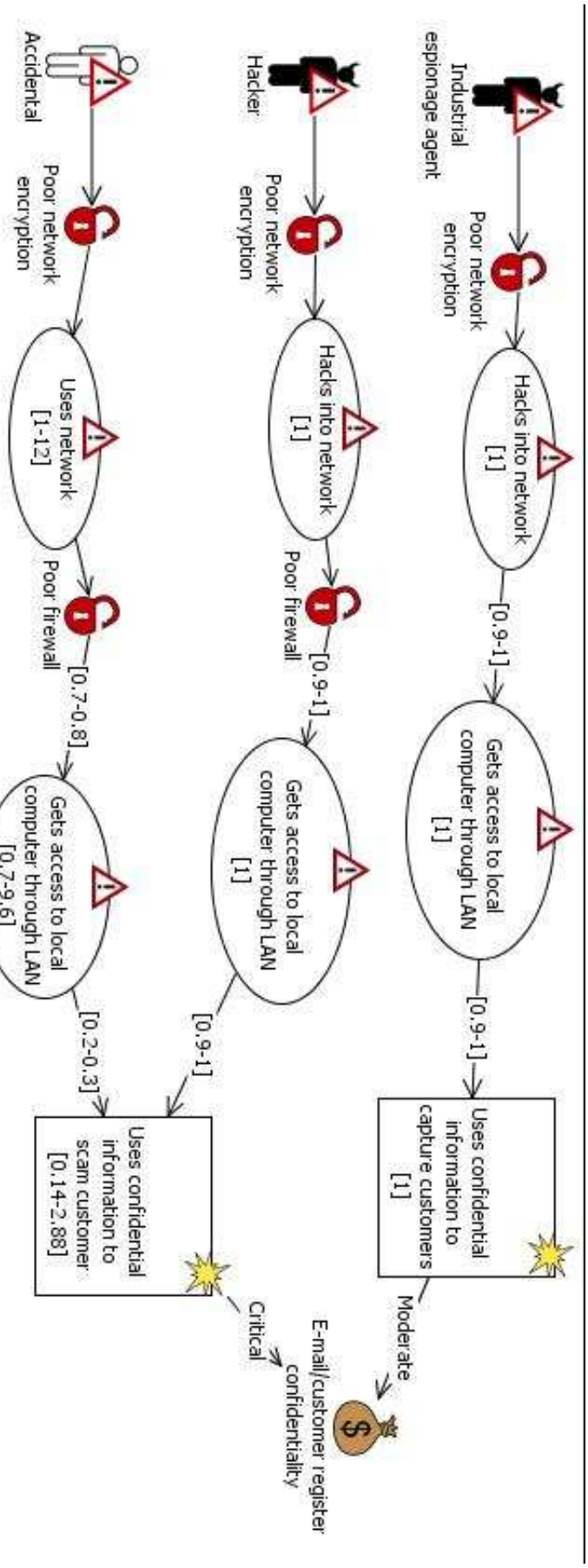

Figure 7.8: E-mail/customer register confidentiality: Risk estimation numerical

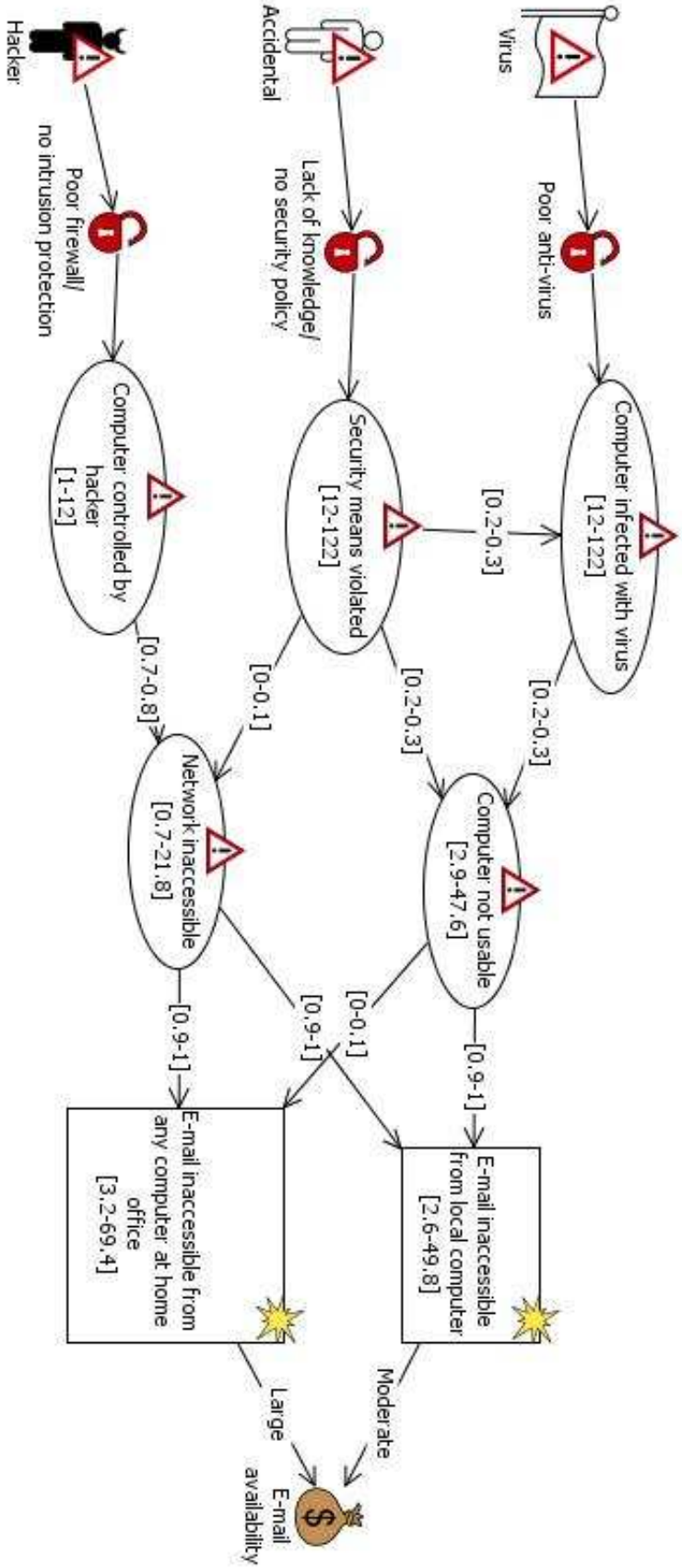

Figure 7.9: E-mail availability: Risk estimation numerical

Defining a set

Sets are defined in the diagram elements as numerical intervals mapped to a linguistic value. Figure 7.10 selects one example that explains how this is represented in the diagram.

Figure 7.10: Defining a set

The threat scenario from figure 7.10 is assigned a frequency value while the initiate relationship has a probability value. Exactly how the likelihood values are assigned, we can see in the table in figure 7.11 which is a screen shot of the likelihood panel in the SCORE tool.

Linguistic	Frequency	Probability
Very low	0-1	<0-0.1]
Low	1-12	<0.1-0.3]
Medium	13-122	<0.3-0.6]
High	123-365	<0.6-0.8]
Very high	365	<0.8-1]

Figure 7.11: Table of likelihood definitions

1-to-1 initiate relationship

The same figure from above (7.10) also illustrates the calculation that has taken place over the initiate relationship from the threat scenario. We see that a frequency of 1-12 times a year yields a slightly lower frequency when its probability to initiate the next threat scenario is as high as 0.7-0.8, namely 0.7-9.6. In order to not loose information through a calculation, we need to keep the exact calculated value in every instance of the calculation through the diagram.

Many-to-1 initiate relationship

Figure 7.12 exemplifies a situation where two threat scenarios both may initiate a third threat scenario. We see that even though one of the threat scenarios has originally a relatively high frequency, it provides little contribution to the aggregated frequency due to its low probability.

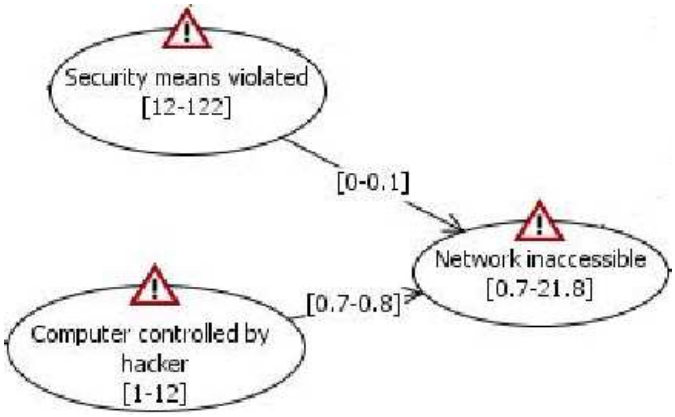

Figure 7.12: Many-to-1 initiate relationship

Set multiplication and addition

To further look into the calculations from the previous example in figure 7.12, the multiplication of the frequency interval $12-122$ and its corresponding probability $0-0.1$ would yield the result $0-12.2$. The other one, $1-12$ to $0.7-0.8$ yields correspondingly the result $0.7-21.8$. Together they are added with the result $0.7-21.8$.

Selecting the correct interval

We continue to use the same example from figure 7.12 and we see that the calculated likelihood do not match or fall withing any of the predefined intervals. The original threat diagram (figure 7.2) however, contains linguistic values, hence we need to decide for one of the intervals that the calculated one shares a part of. The rule 4.3.4 yields to select the interval in which the mean of the calculated interval falls within. In this case, that would be 11.25 that belongs to *Very seldom* as also in the diagram 7.9.

Inconsistencies

Figure 7.13 illustrates a scenario where a threat scenario has been assigned a likelihood value manually by the field expert and then gets initiated by another threat scenario (what we call automatic likelihood assignment). For such incidents, the rule 4.3.5 yields that the combined likelihood should be a calculated result of the intersection of the two contributions instead of the union as in normal cases.

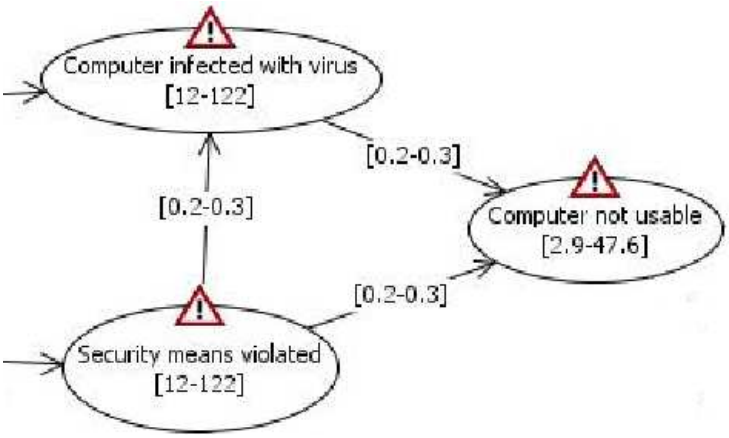

Figure 7.13: Inconsistencies

The sjakk-shop field trial was however not able to test this scenario because the feature was not implemented in the CORAS tool.

Summary

We have illustrated examples of how both numerical frequency and probability sets may be defined. These have further been aggregated through the diagrams by applying rules for both 1-to-1 and many-to-1 initiate relationships that in turn apply set addition and multiplication. Finally mappings from calculated numerical intervals to their correct linguistic expressions were described. The inconsistency rules could however not be illustrated.

The SCORE tool rules provides definitions on how to define the probability sets and linguistic values, but not likelihood values expressed as frequencies. Despite this limitation, we have made the assumption that frequencies can be treated similarly, hence we perform calculations on a mix of probability and frequency representations.

7.3 The SCORE tool testing

In order to test the SCORE tool to its extreme, we perform a software test that serves the purpose of the laboratory experiment proposed in chapter 3 under the evaluation strategies section. The purpose of the testing is to provoke unexpected behaviour.

We create a few specially designed test scenarios that we model using CORAS threat diagrams supported by SCORE:

1. 1-to-1 complete relationships (numerical)
2. Many-to-1 complete relationships (numerical)
3. 1-to-1 complete relationships (linguistic)
4. Many-to-1 complete relationships (linguistic)
5. Interval selection for linguistic representation
6. Inconsistencies (manual)
7. Inconsistencies (automatic)
8. Warnings

The test scenarios are evaluated according to whether or not they exist as a feature in the SCORE tool and if they do; are they fully functional? The complete results and all test scenario listings can be found in appendix D, but below we provide a schematic view of the overall results. Each of the test scenarios is evaluated with respect to whether they meet their respective requirements or not. They are graded as ✓ – met, + – partly met, and - – not met.

Scenario	Exists	Functional
1	✓	✓ not robust
2	+ not constrained to probabilistic domain	✓ not robust
3	✓	✓ not robust
4	✓	✓ not robust
5	✓	✓
6	- not implemented	-
7	- not implemented	-
8	+ partly implemented	-

Table 7.6: The SCORE tool test results

The evaluation of the test scenarios tells us that most of the software requirements are met. Those that fail, 2, 6, 7 and 8, are in general features to improve usability

of the SCORE tool. On the other hand, the acknowledged ones are those vital to prove the success criteria as we will see in the next chapter. There are also issues related to robustness of the SCORE tool in general. Robustness is important when the tool moves on to its next phase of being a commercial implementation, but is not at high priority in a prototype that aims to illustrate functionality and features.

Chapter 8

Fulfilment of success criteria

In this chapter we will evaluate the SCORE method and tool with respect to its defined success criteria and identified requirements. The detailed success criteria are defined in chapter 3 and structured in three parts that we here successively will examine one by one. The evaluation is based on the findings from the evaluation chapter (chapter 7). Each and one of them are validated or falsified during a systematic walkthrough where we provide a short reasoning and point out where they are presented and evaluated. We start with the SCORE method; follow up with the SCORE tool before we lastly examine the SCORE tool-supported method in total with respect to CORAS. Once the detailed success criteria are fulfilled, we use these to argue whether the overall success criteria **P1** - **P5** are fulfilled. Finally this yields enough information in order for us to give a statements on the overall hypothesis **H1**.

During all of the validations, we use the following notations to express degree of fulfilment: ✓ – fulfilled, + – partly fulfilled and - – not fulfilled (or falsified). In cases where the two latter are used, we should provide an explanatory comment and take special consideration to these when further evaluating the overall success criteria.

8.1 Detailed success criteria

8.1.1 The SCORE method

The SCORE method shall: provide rules that defines the risk estimation calculations in CORAS diagrams

1. ***define rules for likelihood calculations on 1-to-1 initiate relationships***

✓ **Fulfilled**

- *calculation of numerical values shall be supported*

→ Rule 4.2.1 and 4.2.3 supports calculation of numerical values

- *calculation of qualitative values shall be supported*

→ Qualitative expressions may be defined as sets according to the rule 4.3.1 and set calculations are defined in rule 4.3.2 and 4.3.3

The 1-to-1 initiate relationship rule is evaluated in 7.2 and found to be fulfilled through logical reasoning. A frequency may only be reduced when a probability no larger than 1 is assigned. The reasoning is acknowledged by both the evaluation referenced, but also the field trial feedback in 7.1.2.

2. ***define rules for likelihood calculations on many-to-1 initiate relationships***

✓ **Fulfilled**

- *calculation of numerical values shall be supported*

→ Rule 4.2.2 and 4.2.3 supports calculation of numerical values

- *calculation of qualitative values shall be supported*

→ Qualitative expressions may be defined as sets according to the rule 4.3.1 and set calculations are defined in rule 4.3.2 and 4.3.3

The many-to-1 initiate relationship rule is evaluated in 7.2 and found to be fulfilled through logical reasoning. When two or more elements initiate another element with respective given probabilities, the aggregated likelihood is a result of the sum of all initiating elements' contributions. The reasoning is acknowledged by both the evaluation referenced, but also the field trial feedback in 7.1.2.

3. ***define how to translate qualitative values into numerical sets applicable for likelihood calculations***

✓ **Fulfilled**

→ Qualitative expressions may be defined as sets according to the rule 4.3.1

Set definitions are evaluated in 7.2 and together with the translation between sets and qualitative expressions evaluated in 7.2, qualitative expressions are proved to possibly be defined as sets.

4. ***define rules for dealing with inconsistencies***

\+ **Partly fulfilled**

\- **Comment: rule exists, but is not sufficiently tested**

→ Rule 4.3.5 and 4.3.5 defines behaviour when inconsistencies occur.

8.1.2 The SCORE tool

The SCORE tool shall: provide a tool-support for the SCORE method

1. ***provide the ability to assign likelihood values to threat scenarios***

✓ **Fulfilled**

- *both numerical and qualitative values shall be available to be assigned*

→ the existence of the feature is proved in 6.2 and further evaluated in 7.3

- *the likelihood values shall be stored in the unique node*

→ the implementation, in 6.4 proves that threat scenarios are extended with an attribute to hold the assigned likelihood, and this is implicitly evaluated through the whole of the field trial (7.1)

2. ***provide the ability to assign likelihood values to unwanted incidents***

✓ **Fulfilled**

- *both numerical and qualitative values shall be available to be assigned*

→ the existence of the feature is proved in 6.2 and further evaluated in 7.3

- *the likelihood values shall be stored in the unique node*

→ the implementation, in 6.4 proves that unwanted incidents are extended with an attribute to hold the assigned likelihood, and this is implicitly evaluated through the whole of the field trial (7.1)

3. ***provide the ability to assign likelihood values to initiate relationships***

+ **Partly fulfilled**

- **Comment: ability exists, but no restrictions on input values provided**

  - *the likelihood values shall be probability values in the range [0-1]*

  → the implementation allows for the probabilities to be assigned a probability value, but yields no constraints on the legal range of the probabilities

  - *the likelihood values shall be stored in the unique relationship*

  → the implementation, in 6.4 proves that initiate relationships are extended with an attribute to hold the assigned likelihood, and this is implicitly evaluated through the whole of the field trial (7.1)

4. ***provide the ability to assign consequence values to harm relationships***

✓ **Fulfilled**

→ the existence of the feature is proved in 6.2 and further evaluated in 7.3

5. ***automatically calculate possible likelihood values according to the SCORE method***

✓ **Partly fulfilled**

- **Comment: feature exists, but is not sufficiently tested with respect to robustness**

→ the existence of the feature is proved in 6.2 and 6.2 and further evaluated in 7.3

  - *the tool shall perform a test to validate that sufficient input data is available upon execution*

  → the test is not fully implemented and robustness of system is only partly tested in 7.3

  - *the tool shall keep the exact calculated values for further calculations in cases of round-offs and translation between numerical and qualitative values*

  → the existence of the feature is proved in 6.4 and further evaluated in 7.3

6. ***provide an overview of the properties of the diagram elements***

✓ **Fulfilled**

→ the existence of the feature is proved in 6.2 and further evaluated in 7.3

8.1.3 The SCORE tool-supported method in CORAS

The SCORE tool-supported method shall: be integrated with CORAS

1. apply to CORAS diagrams

✓ **Fulfilled**

- the tool-supported method shall apply to the risk estimation phase and threat diagrams in special

→ this criteria is proved in the field trial where only CORAS threat diagrams are used to assign likelihoods (7.1)

2. be integrated with the CORAS tool

✓ **Fulfilled**

→ the screen-shot documentation in 5.1 is a living proof that the SCORE tool is integrated withing the CORAS tool and so also with the implementation documentation (6.4)

8.1.4 Summary

The general impression from the validation above is that most of the success criteria are fulfilled. The SCORE method and the tool-supported method's integration in CORAS score the best results. The SCORE tool is where most problems occur, but still none of the criteria are falsified. However, this does not mean that the tool is fully functional – there is obviously a lot of work to be done, especially on robustness issues to make the tool unassailable.

8.2 Overall success criteria

Based on the evaluation of the detailed success criteria above, we argue the fulfilment of the overall success criteria which are dependent on the results from above.

A carefully designed tool-supported method:

Overall success criterion 1

P1: *will provide the ability to merely assign likelihood values to threat scenarios and automate likelihood aggregation to unwanted incidents*

✓ **Fulfilled**

The field trial (7.1) only assigned manual likelihood values to threat scenarios and used the tool-supported method to calculate the likelihoods to the unwanted incidents. Based on this and the validations from above, we also validate **P1** as fulfilled.

Overall success criterion 2

P2: *will be able to calculate likelihood values for diagram elements initiated through intricate relations*

✓ **Fulfilled**

To account for *intricate relations* first, we assume this to apply to *many-to-1 initiate relationships* and even reiterated occurrences of such. Both the SCORE method and the SCORE tool have been validated for the possible relations and in that respect we claim **P2** to be validated as fulfilled accordingly.

Overall success criterion 3

P3: *will handle an arbitrary large number of input*

\+ **Partly Fulfilled**

The SCORE method applies no restrictions on the number of input and the rules are confirmed to be valid through no matter how much input we provide. The SCORE tool however, has been proved to suffer under robustness issues, hence cannot guarantee for an arbitrary large number of input. Due to this, **P3** is only partly fulfilled.

When concerning effectiveness in terms of time-saving, the tool-supported method should apply means to what type of input to process.

Overall success criterion 4

P4: *will provide the ability to merely deal with qualitative data, but still yield calculations thereupon*

✓ **Fulfilled**

Mappings between qualitative data and numerical intervals have been defined to account for this and the tool is designed with the same purpose. Both set

mappings and set calculations have been validated above, hence we may also validate **P4** as fulfilled.

Overall success criterion 5

P5: *will provide the ability to merely assign likelihood values to threat scenarios and automate likelihood aggregation to unwanted incidents*

✓ **Fulfilled**

This prediction yields the same criteria as **P1** except for the time-saving perspective. In that respect we need to look more into usability of the SCORE tool. It is obvious that a tool that may break down or yield unexpected behaviour may cause major time loss. Although the SCORE tool suffers under robustness issues, there is little doubt that a *carefully designed* and hence robust tool would validate **P5** as fulfilled with the reasoning from **P1** kept in mind and time-saving accounted for. Therefore **P5** is also fulfilled.

8.3 Overall hypothesis

H1: *a carefully designed tool-supported method will increase the efficiency of risk estimation using CORAS diagrams*

From the evaluation of both the overall and detailed success criteria, we have seen that most of the criteria are fulfilled and just a few partly fulfilled. None has been falsified, and the most severe problems have been related to the implementation of the SCORE tool and the robustness of this. However, the SCORE tool is meant to be a prototype with the purpose to illustrate the functionality of a computerised tool that we may use to evaluate the overall hypothesis **H1**. We believe that both the SCORE tool and the SCORE method have fulfilled their mission, and if we assume that the overall success criteria together with the detailed success criteria covers sufficient aspects of the overall hypothesis to be able to validate this, we may validate **H1** and claim **H1** to be fulfilled.

Chapter 9

Discussion

In the precedent chapter we have seen the success criteria from chapter 3.6 been validated with respect to our research and evaluation strategies. In this chapter we account for assumptions and considerations and possible threats to validity of our results, with the purpose to reveal eventual unexplained causes that could possibly affect our findings enough to question the liability of the SCORE project. We do this by first looking into assumptions and considerations made along the way. Then we investigate the validation of the success criteria and what we see as threats for hence. At last we perform a final validation of the success criteria again based on all of the above.

9.1 Assumptions and considerations

During the SCORE project work there have been several assumptions and considerations we have had to make. Some of them well argued and discussed, but also those that have come naturally or have been necessary to decide for ad-hoc. First we will look into our selection of background material and see if this is sufficient enough for our problem description. Next we review the proposed and later executed research methods. In the following comes an examination of aspects concerning the SCORE method before the SCORE tool subsequently. At last we discuss aspects related to the evaluation of SCORE.

9.1.1 Selection of background material

The background material provided concerns in general security risk analysis and different technologies and methods used to perform such. Most importantly is the presentation of CORAS diagrams. This is obviously inevitable due to the focus of the SCORE project, but the other established techniques are results of a deliberate selection.

The selection is inspired by the needs for the CORAS risk estimation phase, amongst others described in [3], but also to a large extent on the NS 5814 guide to risk analysis [6]. The latter also suggests an analysis technique called FMECA (failure modes, effects and criticality analysis) that could have been relevant for SCORE. FMECA aims to identify potential system failures, but is however not directly applicable to SCORE. It may rather be used as a preliminary analysis technique whose results may constitute a foundation for our selected techniques to build on.

One objective of SCORE is to deal with qualitative data received as input from the risk estimation phase. The SCORE method applies basically mathematical theories to handle such input. However there are research areas that have examined those problems on a large scale, often called fuzzy theories. Research of Papadaki [19] [20], amongst others [21], proposes theories containing rather complex algorithms for computation of likelihood values. Most of the theories are based on input from multiple experts, and we have used the theories as inspiration in our work with the SCORE method. However, we have come to the conclusion that we need to determine if the basic ideas of SCORE are worth implementing in the first place. Fuzzy theories may hence be proposed as a future extension of SCORE.

9.1.2 Chosen research techniques

In chapter 3 we propose a few research strategies that the SCORE project should follow both with respect to requirements retrieval and evaluation of the developed tool-supported method. We argue that our research strategies should fulfil McGrath's three characteristics [28], but there would always be room for performing more and better strategies to document either or. What limit the possible number of such techniques is often time and available resources, so also with the SCORE project. Below, in section 9.2.2 we look into one technique that could preferably have been conducted in the SCORE project, namely a field trial similar to the initial case study.

9.1.3 Designated SCORE method rules

The SCORE method rules defined in chapter 4 evolved from the needs of SCORE we identified in the preceding chapter 3. Again, related to the discussion about the research techniques in the previous section, we could have applied qualitative in-depth interviews with current CORAS experts and users at SINTEF to be absolutely certain that the correct needs have been captured. Such research would also probably further expanded the user requirements for SCORE.

At the time of completion of the case study, we believed (and still do) that more than enough background material was recovered and sufficient experience gained in order to state the requirements for SCORE.

9.1.4 Design of the SCORE tool

Upon modelling the architectural design of the SCORE tool, we had to make a decision on to what extent the SCORE tool should be a stand-alone supportive tool to the CORAS tool or an integrated part of this. The problem is addressed also in chapter 3, and we conclude that in order to be applied to a CORAS security risk analyses in practise, it is a necessity for the SCORE tool to be an integrated part of the CORAS tool.

How to visually represent SCORE and how to provide the interface for data collection (i.e. assignment of likelihood values) plus display the calculation results have also been subject to discussion. The focus of the project is however not usability or graphical design, hence there exist little research on this. What became obvious during the development was first of all that the likelihoods should be visible *on* the CORAS diagram elements. The bottom panel containing properties and likelihood definitions etc. is not that obvious, but is inspired by other modelling tools such as the eclipse development platform [33]. A panel like that is easily extendable to contain other features not related to SCORE such as attributes and properties, and offers an easy way to both inspect and alter the information provided.

9.1.5 Application of SCORE in the evaluation

During the sjakk-shop field trial we made an important assumption related to legal definitions of likelihood values. Likelihood is defined in the CORAS language [25] as a general description of *frequency* or *probability*. The SCORE method only defines probabilistic sets as legal likelihood representations, but since they

are hard to relate to as frequencies of incidents, we carried through with calculations also applied to frequency representations. The initiate relationships were probability representations though, thus the factual calculations were executed using a mix of both representation types. A dedicated evaluation of probability calculations exclusively may be investigated in the experimental simulation in chapter 7.3 instead.

Even though the tool-supported method provides both logic's and tool-support for both frequency and probability representations, it does **not** imply that this is a part of the CORAS language and thus defined in the CORAS semantics. To determine this would be an own project itself and is up to the respective experts of CORAS and beyond the scope of SCORE. However, this should be looked upon as a proposal to further research in the development of CORAS.

9.2 Threats to the validity of results

In all research there may exist uncertain variables affecting the results of the research. Here we will try to identify what could possibly threaten the validity and liability of our findings by criticising our executed research methods.

9.2.1 Quality assurance of SCORE requirements

Chapter 3 defines the success criteria for the SCORE project and chapter 5 defines the requirements to the SCORE tool. They are all based on research from a case study (section 3.2) and findings from the problem analysis in chapter 3 that both aim to identify the needs for the SCORE tool-supported method. However, they are never subject to re-examination or quality assurance before they are implemented. In order to play by the book, requirements should be evaluated before implementation to make sure that we in fact have captured and fulfilled the actual needs. We may have convinced ourselves that we have done so, but such requirements should be formally approved by the client commissioning the project.

9.2.2 Liability of field trial

The risk analysis performed as the field trial for realistic evaluation of the SCORE tool-supported method was performed in a subjective manner. The risk analysis leader was this master thesis's candidate and the field expert (and target owner)

a family member of his. Such close relations to the results of the field trial could of course affect their objectiveness since they would both care for the final results of the master thesis.

Preferably, the risk analysis should be performed by a professional risk analysis team with no intervention or subjective interest in the development of the SCORE tool-supported method. A field trial or case study similar to the initial case study performed to identify the needs for SCORE (refer to 3.2) would have been perfect in that respect. The field trial would involve SINTEF and require resources from their current research project that involves CORAS. Such study would unquestionably provide the best realism since it then would be tested and examined in its natural and ever intended application environment.

Although CORAS is constantly developing and all field trials are run with the purpose to evaluate and improve the various aspects of CORAS, we cannot forget the client and customer perspective regarding such security risk analyses. As cooperative partners, the commissioning clients have their own interests in the security risk analysis. They put a lot of effort and resources into conducting a security risk analysis within their organisation and hence demand quality and professionalism from the risk analysis team and their methods. In that respect; SCORE should been proved as an advantageous tool-supported method worthwhile implementing in the first place before introduced as a dedicated part of CORAS. This is what we aim to achieve with the *sjakk-shop* field trial and we believe it has fulfilled its purpose.

9.2.3 Research strategies – empirical investigations

Motivated by the statement from the previous section, we should preferably have tested the SCORE tool-supported method on a liable industrial risk analysis. With such a risk analysis available, we could have produced empirical data for validation and compared these to the initial case study. Empirical data from in-depth interviews or field observations provides particular realism to a research study [28].

9.2.4 Liability of experimental simulation

The test scenarios we have created for the SCORE tool were obviously too few and less comprehensive. Also, the chosen test sets should have been an arbitrary large number of random sets.

9.3 Validation

Finally in this chapter we will answer the question:

> *Are the impact of the identified threats to the validity of results and our assumptions and considerations significant enough to affect the total liability of the success criteria validation?*

Based on the discussions above, we conclude that the liability to the evaluation of SCORE has the most severe reason to be questioned, but still the validation of our superior hypothesis remains liable. However, further testing of the SCORE tool would be a natural part of any software engineering project and hence subject to further development. The SCORE method must also continue evolving over time from experiences made from applied field trials.

Chapter 10

Conclusion

This thesis has presented and documented the results from the research of the SCORE project. The SCORE project has successfully investigated the hypothesis:

H1: *a carefully designed tool-supported method will increase the efficiency of risk estimation using CORAS diagrams*

Hypothesis **H1** aims to handle problems related to documenting and processing risk estimations using CORAS diagrams. The needs for such method evolve from the risk estimation phase of a CORAS security risk analysis where the participating parties often experience difficulties concerning risk estimations. The main objective of the project has been to develop a tool-supported method to overcome this troublesome activity. Success of the SCORE project has been evaluated with respect to **H1** through validation of defined success criteria.

*The SCORE tool-supported method has in total been evaluated as a success with respect to its overall hypothesis **H1**.*

The SCORE project has contributed with a well-documented tool-supported method for risk estimation using CORAS diagrams; the SCORE method and tool. Based on evaluation of our proposed tool-supported method, among them a CORAS security risk analysis, we believe that the SCORE method and tool should be implemented with the next official release of the CORAS tool and such be applied to future CORAS security risk analyses.

10.1 Findings

The following lists the main findings of the SCORE project:

The SCORE project has:

- revealed a need for a structured methodical approach to the activity of risk estimation in a CORAS security risk analysis based on: a case study of such, investigation CORAS diagrams, and analysis of established risk modelling techniques
- identified a method providing a set of rules with the purpose to structure and standardise the documenting and processing of risk estimation data; the SCORE method
- evolved a computerised tool prototype providing support for the SCORE method and an interface to document and automate the processing of risk estimation data; the SCORE tool
- been evaluated as a success with respect to its overall hypothesis **H1**

10.2 Future work

From our experiences through working with the SCORE project we have experienced what could be further work related to risk estimation in CORAS diagrams.

10.2.1 The SCORE method

The SCORE method can be developed further in several directions. Primarily, there are techniques related to fuzzy theories that may be applied.

During a risk analysis there may be multiple system experts present with different opinions when it comes to defining likelihood and consequence values. Instead of forcing these to agree on a compromise value, we may calculate a combined value based on each individual statement. This may especially come to hand when the risk analysis workshops are organised with teamwork from multiple teams. We recommend to apply the theories from Papadaki: fuzzy multiple criteria decision making method [19] and [20].

10.2.2 The SCORE tool

The SCORE tool needs extensive work with respect to usability in order to be applicable for the next official release of the CORAS tool. The implementation needs to be improved such that it reaches a commercial level and not the state of a prototype as for now. Furthermore should the requirements specifications be investigated and be asserted to be implemented fully.

Bibliography

[1] IBM Internet Security Systems X-Force research and development team. IBM internet security systems x-force 2006 trend statistics. Technical report, IBM Corporation, 2007. [cited at p. 1]

[2] Ida Hogganvik and Ketil Stølen. Empirical investigations of the CORAS language for structured brainstorming. Technical report STF90 A05041, SINTEF ICT, 2005. [cited at p. 1]

[3] Folker den Braber, Ida Hogganvik, Mass Soldal Lund, Ketil Stølen, and Fredrik Vraalsen. Model-based security analysis in seven steps – a guided tour to the CORAS method. Technical report, SINTEF ICT, 2007. [cited at p. 1, 104]

[4] Iselin Engan, Mass Soldal Lund, and Stig Torsbakken. Evaluations of the method and tool used during the 8th SECURIS field trial. Technical report, SINTEF ICT, 2007. [cited at p. 2, 26, 28]

[5] David F. Haasl. Advanced concepts in fault tree analysis. In *Proceedings of System Safety Symposium*, Seattle, Washington, 1965. [cited at p. 7, 9, 10, 12]

[6] Marvin Rausand. *Risiko Analyse - Veiledning til NS 5814*. Tapir, 1991. [cited at p. 7, 9, 10, 13, 31, 104]

[7] ISO/IEC 14977:1996(E). Information technology – syntactic metalanguage – extended bnf. Standard 14977:1996(E), ISO/IEC, 1996. [cited at p. 8]

[8] Australian/New Zealand Standard AS/NZS 4360:2004:. Risk management. Standards, Standards Australia (SA), 2004. [cited at p. 8]

[9] IEC 60300-3-9. Dependability management – part 3: Application guide – section 9: Risk analysis of technological systems. Standards, IEC, 2004. [cited at p. 8]

[10] Bruce Schneier. *Secrets & lies: Digital Security in a Networked World*. John Wiley & Sons, 2004. [cited at p. 15, 16]

[11] Ida Hogganvik. *A Graphical Approach to Security Risk Modeling*. PhD thesis, University of Oslo, 2007. Unpublished. [cited at p. 18, 21]

[12] ISO/IEC. Information technology - syntactic metalanguage - Extended BNF. Technical report 14977:1996(E), ISO/IEC, 1996. [cited at p. 19]

[13] Ida Hogganvik and Ketil Stølen. Specification of the language including modelling guidelines. 2006. [cited at p. 20, 31, 34]

[14] Heidi E. I. Dahl, Ida Hogganvik, and Ketil Stølen. Structured semantics for the CORAS security risk modelling language. Technical report, SINTEF ICT, 2007. [cited at p. 21, 32, 39]

[15] Ida Solheim and Ketil Stølen. Teknologiforskning - hva er det? Technical report STF90 A06035, SINTEF ICT, 2007. [cited at p. 27, 35, 36]

[16] CORAS - a platform for risk analysis of security critical systems, September 2003. http://coras.sourceforge.net/. [cited at p. 28, 30]

[17] SECURIS - model-driven development and analysis of secure information systems, 2007. http://www.sintef.no/content/page1___1824.aspx. [cited at p. 28]

[18] DIGIT - digital interoperability with trust, 2007. [cited at p. 28]

[19] Katerina Papadaki. A fuzzy multiple criteria decision making method for the selection of information security measures. Technical report, National technical university of Athens, 2005. [cited at p. 28, 34, 104, 110]

[20] Katerina Papadaki. A fuzzy methodology for assessing the aggregate impact of security incidents. Technical report, National technical university of Athens, 2005. [cited at p. 28, 104, 110]

[21] Fang Liu, Kui Dai, Zhiying Wang, and Jun Ma. Research on fuzzy group decision making in security risk assessment. Technical report, School of Computer, National University of Defense Technology, China, 2005. [cited at p. 28, 104]

[22] Merriam-Webster online dictionary, 2007. http://www.m-w.com. [cited at p. 29]

[23] Scott Plous. *The psychology of judgment and decision making*. McGRAW-HILL, 1993. [cited at p. 29]

[24] CORAS, September 2003. `http://coras.sourceforge.net/`. [cited at p. 30]

[25] Ida Hogganvik and Ketil Stølen. A graphical approach to risk identification, motivated by empirical investigations. Technical report, SINTEF ICT, 2006. [cited at p. 31, 34, 105]

[26] U.S. Environmental Protection Agency. Technical guidance for hazards analysis – emergency planning for extremely hazardous substances. Technical report, U.S. Environmental Protection Agency, 1987. [cited at p. 32]

[27] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Bjöorn, Regnell, Anders Wesslén, and Victor R. Basili (editor). *Experimentation in Software Engineering: an Introduction*. Kluwer Academic Publishers, 2000. [cited at p. 35, 36]

[28] James E. McGrath. *Groups: Interaction and Performance*. Prentice-Hall, 1984. [cited at p. 36, 104, 107]

[29] Seymour Lipschutz and Marc Lipson. *Schaum's outline of theory and problems of discrete mathematics*. McGRAW-HILL, second edition, 1997. [cited at p. 39]

[30] Ian Sommerville. *Software engineering*. Pearson Addison Wesley, seventh edition, 2004. [cited at p. 51]

[31] The SCORE tool, 2007. http://folk.uio.no/stigt/score/. [cited at p. 55]

[32] The CORAS tool subversion repository, September 2003. https://coras.svn.sourceforge.net/svnroot/coras. [cited at p. 70]

[33] Eclipse SDK, 2007. http://www.eclipse.org. [cited at p. 105]

Appendices

Appendix A

The SCORE tool architectural design

In this appendix we present the architectural design of the SCORE tool. For textual descriptions; refer to chapter 6. Note that the interaction diagrams are not fully complete, but is subject to black-boxing of the most implementation specific details.

The SCORE tool architectural design

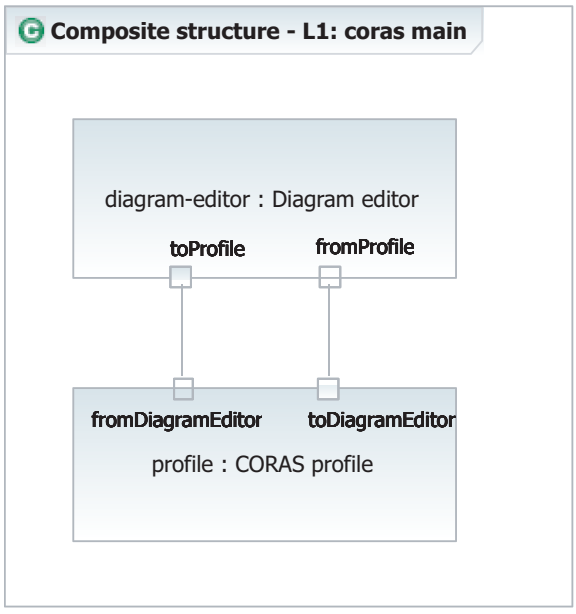

Figure A.1: Level 1 composite structure diagram

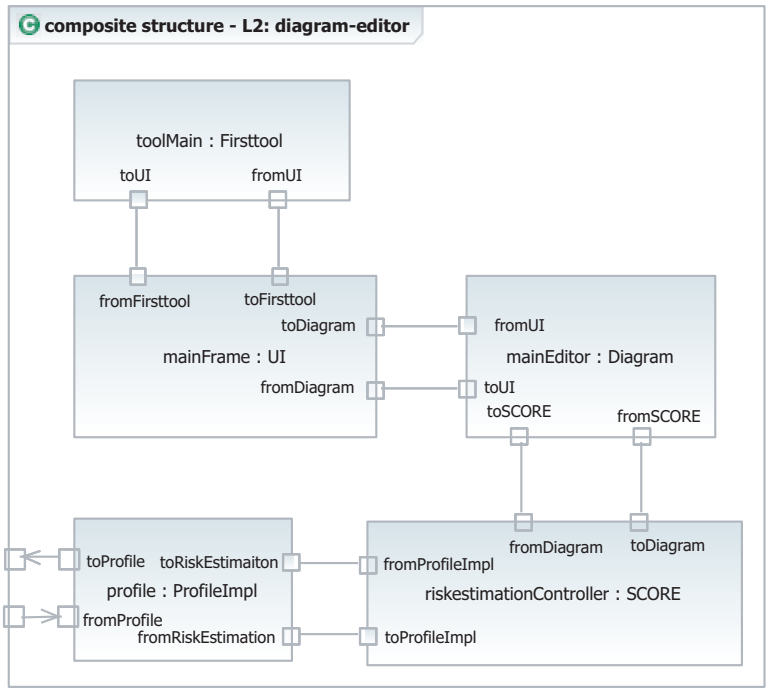

Figure A.2: Level 2 composite structure diagram

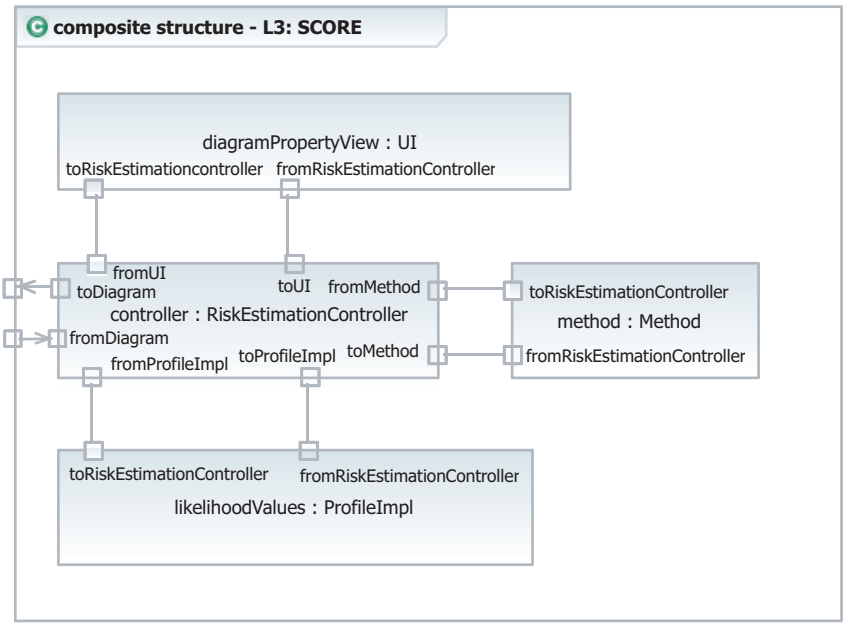

Figure A.3: Level 3 composite structure diagram

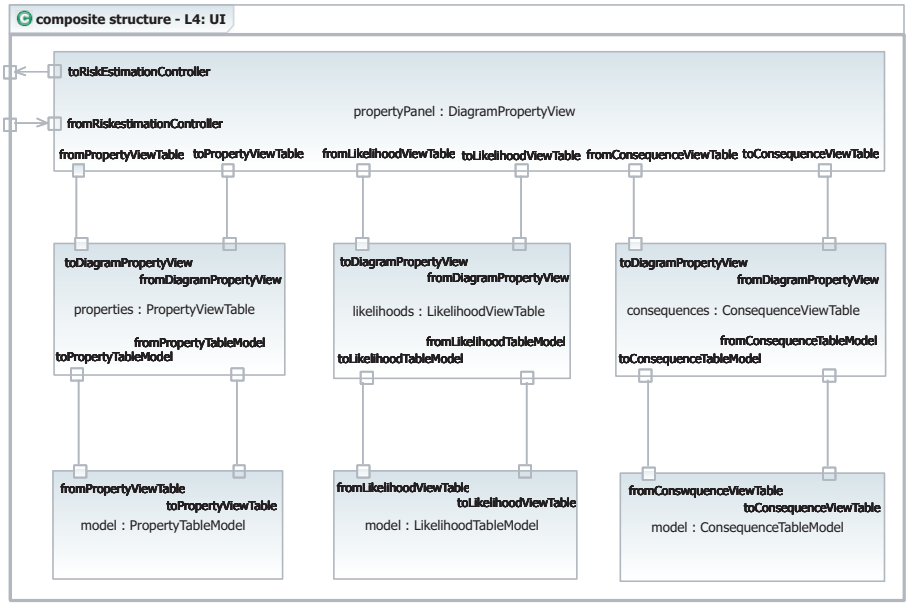

Figure A.4: Level 4 composite structure diagram

The SCORE tool behavioural design

Here we provide interaction diagrams for the use cases: *calculate likelihood value* and *assign likelihood value*.

Calculate likelihood value

This use case is initiated by the user when likelihood values have been assign to the diagram element's initiating elements and relations. We only describe a *happy-day* scenario and do not account for negative behaviour.

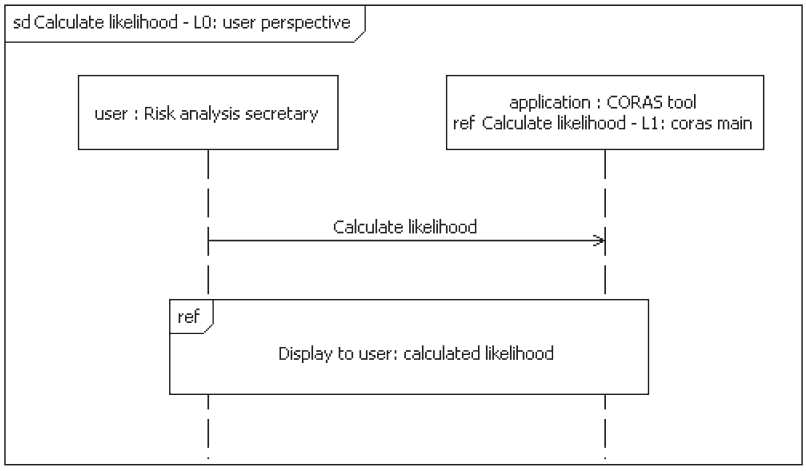

Figure A.5: Level 0 interaction diagram - calculate likelihood value

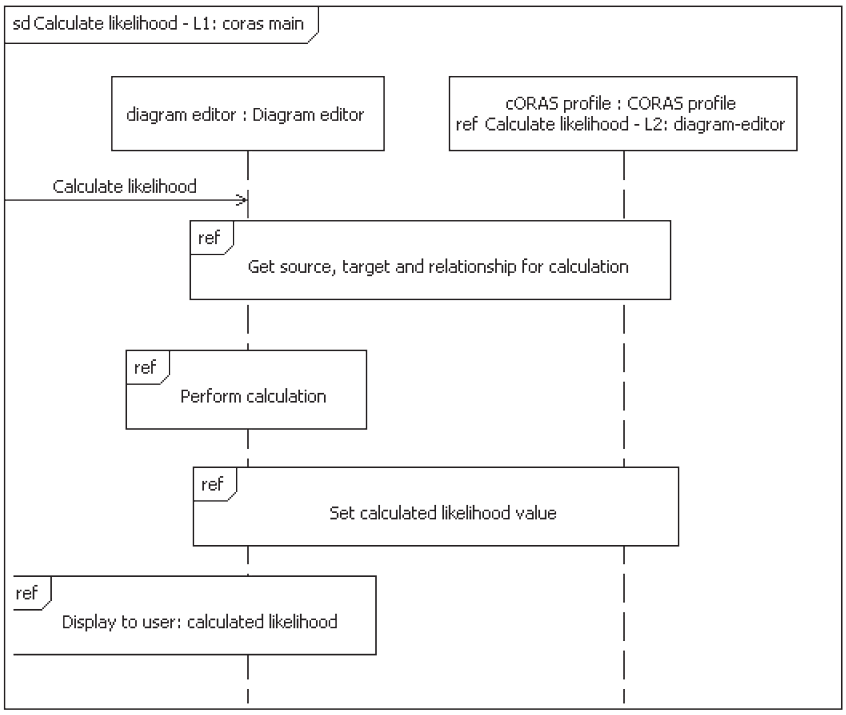

Figure A.6: Level 1 interaction diagram - calculate likelihood value

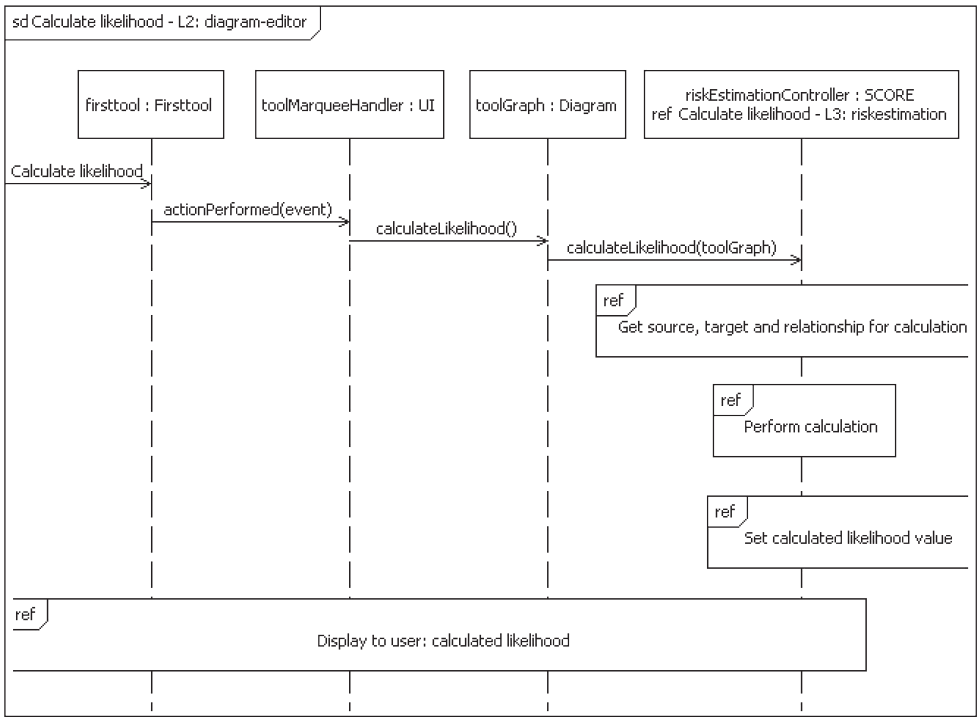

Figure A.7: Level 2 interaction diagram - calculate likelihood value

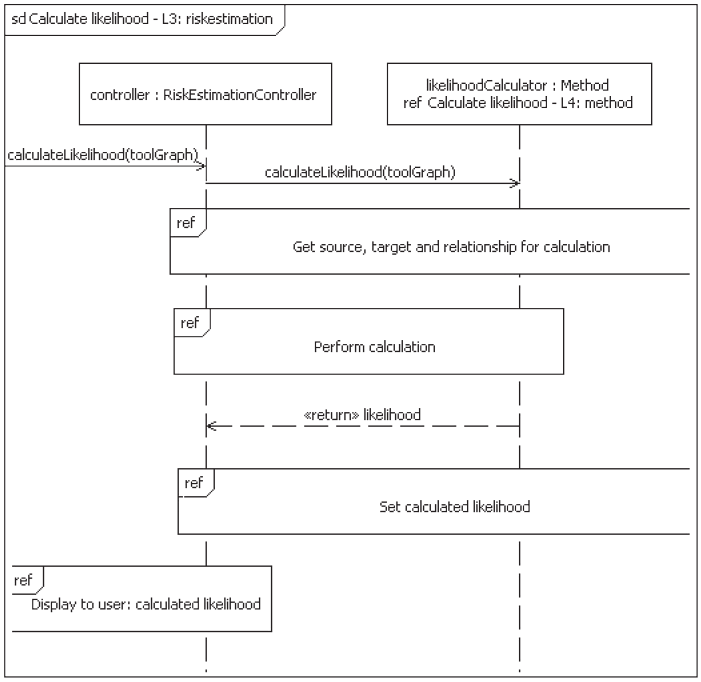

Figure A.8: Level 3 interaction diagram - calculate likelihood value

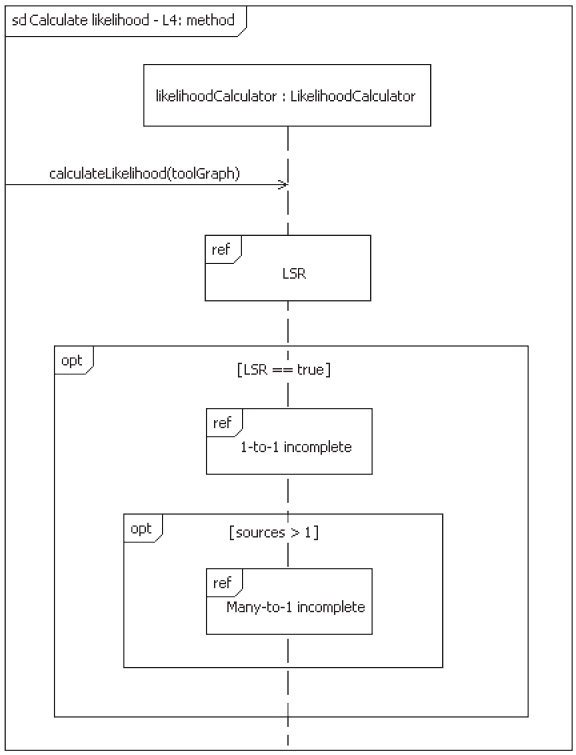

Figure A.9: Level 4 interaction diagram - calculate likelihood value

Assign likelihood value

This use case describes the system behaviour when a user tries to assign a likelihood value to a CORAS diagram element. Note that this description must be considered as deprecated since there has been some adjustments with respect to its functionality. This use case automatically calculates likelihood values upon assignment, while this has later been changed to an own use case initiated by the user.

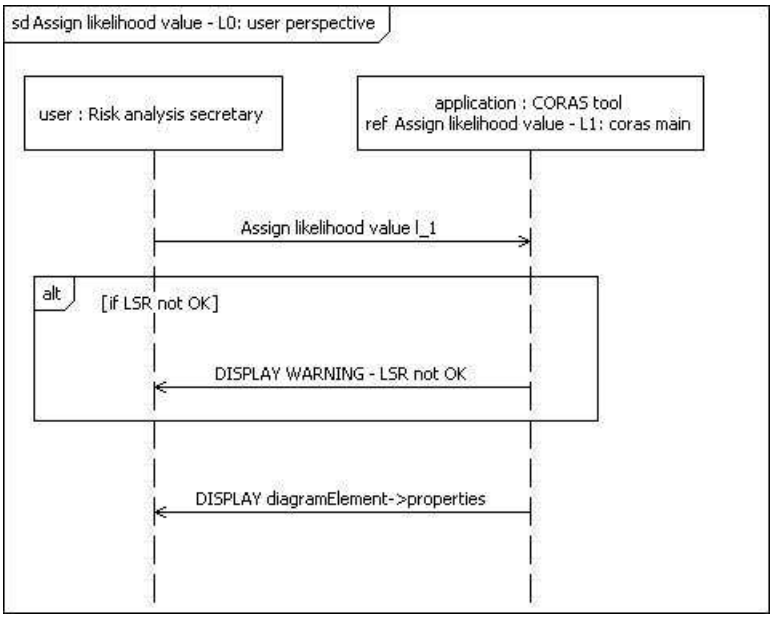

Figure A.10: Level 0 interaction diagram - assign likelihood value

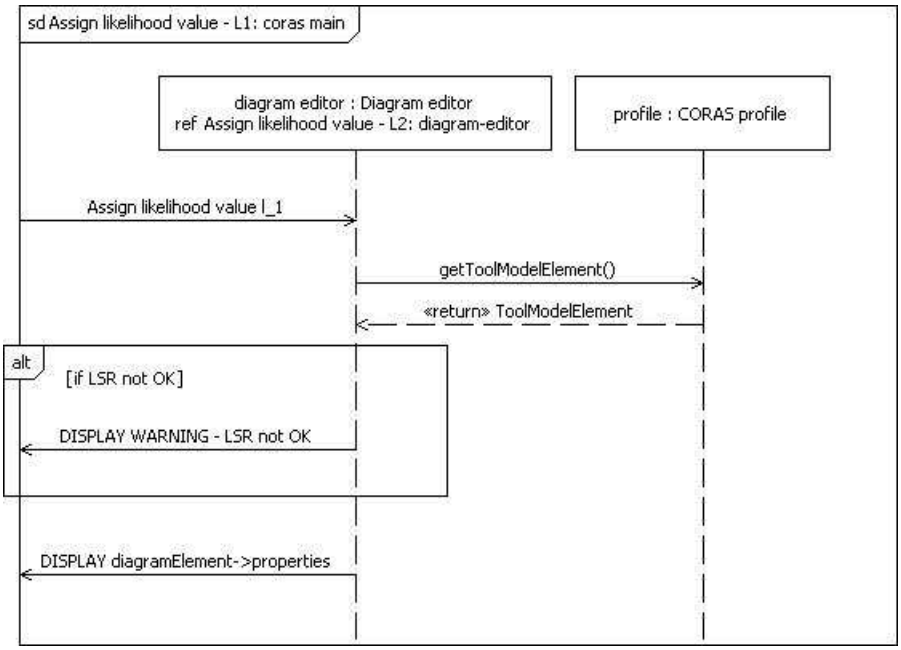

Figure A.11: Level 1 interaction diagram - assign likelihood value

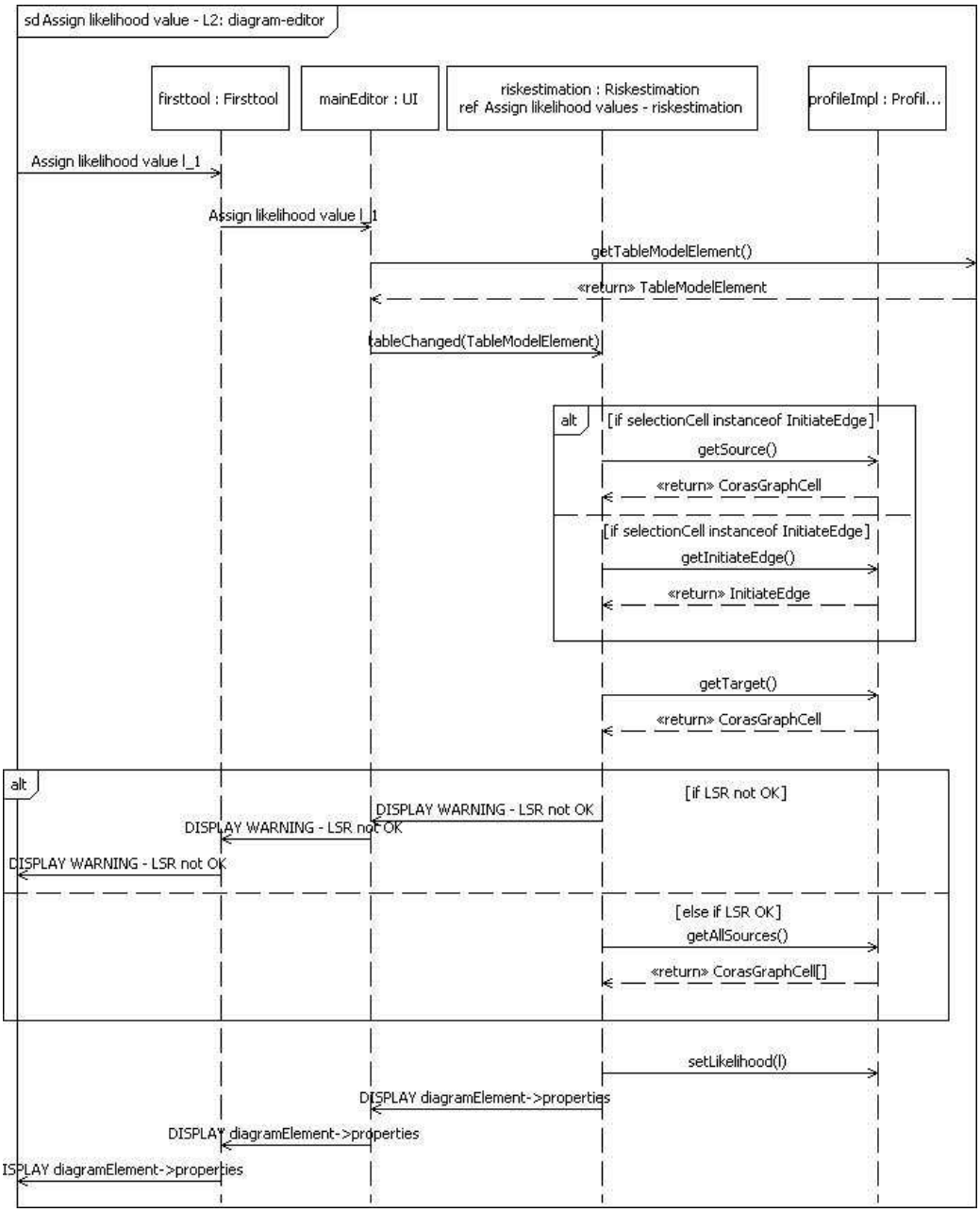

Figure A.12: Level 2 interaction diagram - assign likelihood value

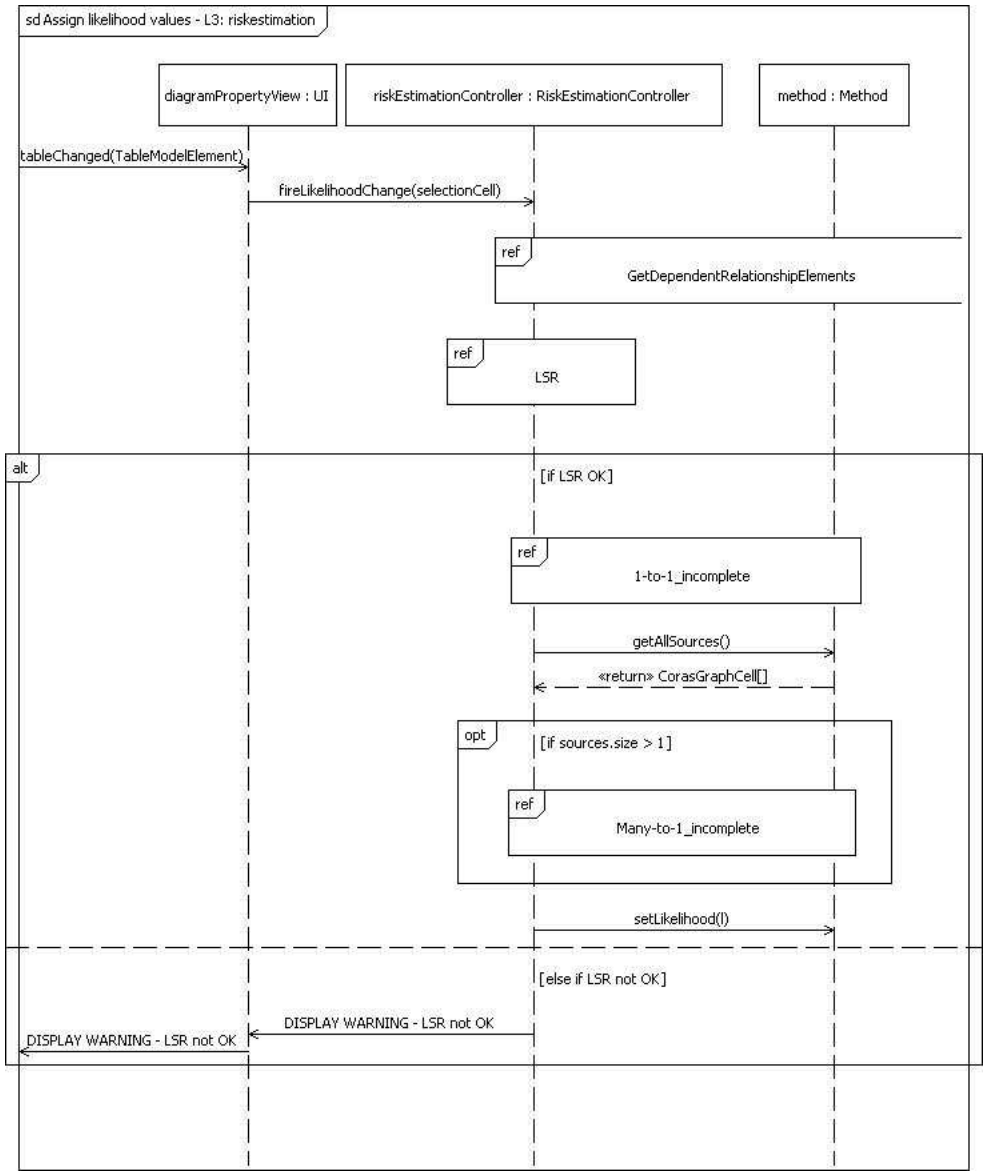

Figure A.13: Level 3 interaction diagram - assign likelihood value

The SCORE tool implementation specific design

Here we provide a total overview of the containing classes of the SCORE tool, and full details of a number of selected classes with their respective attributes and operations.

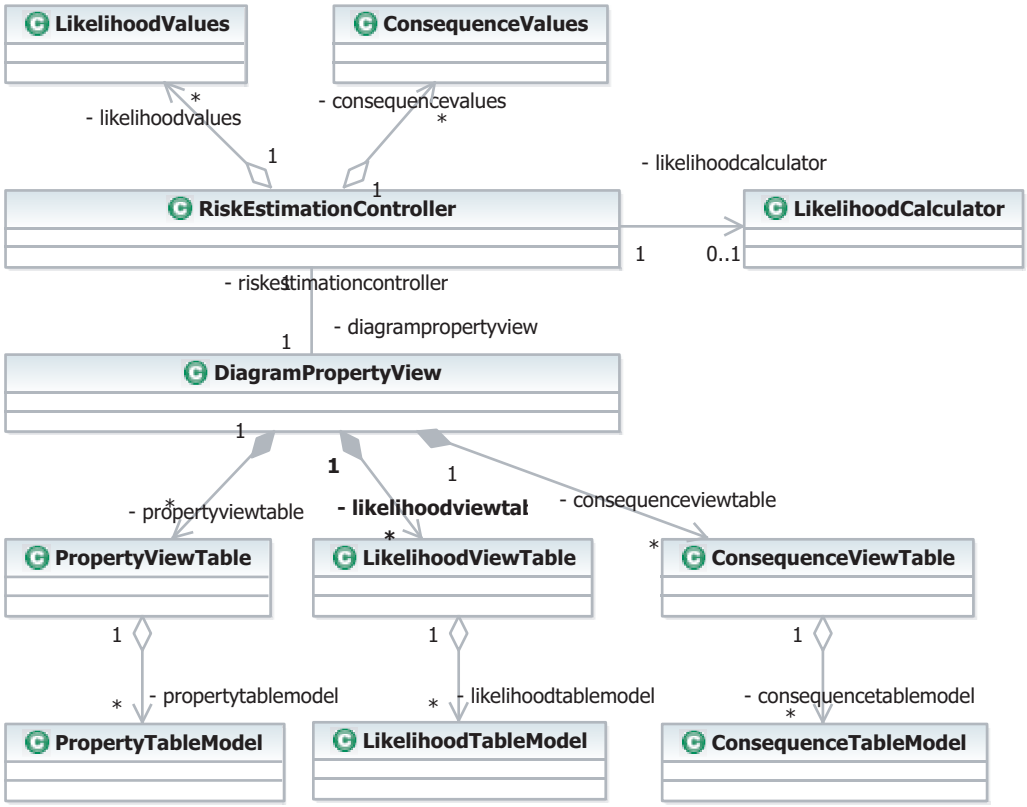

Figure A.14: The SCORE tool overview class diagram

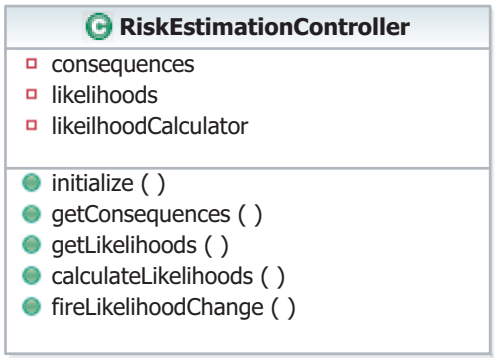

Figure A.15: The SCORE tool main controller class

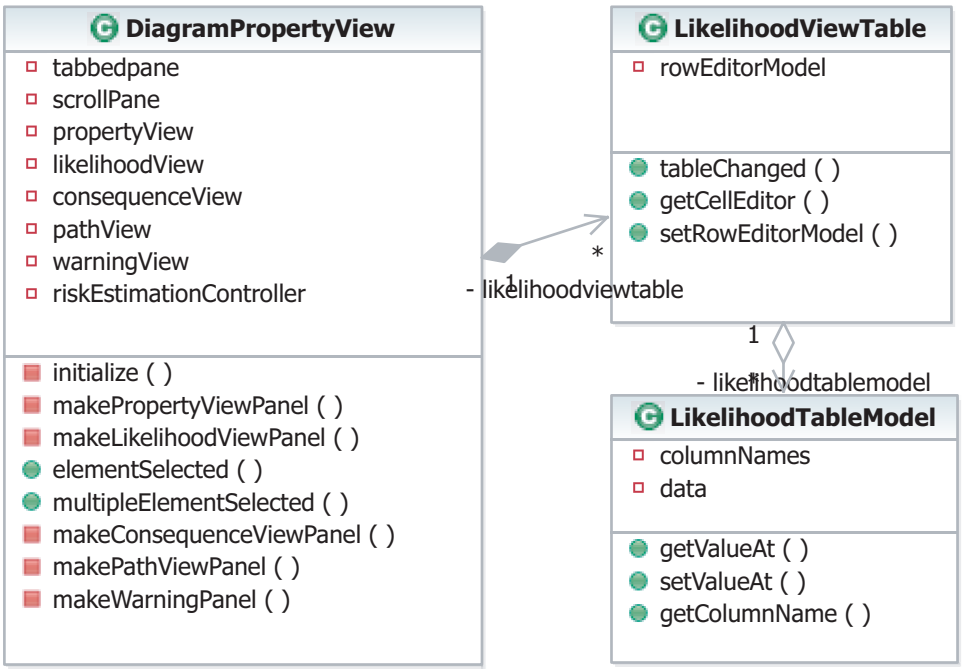

Figure A.16: The SCORE tool user interface

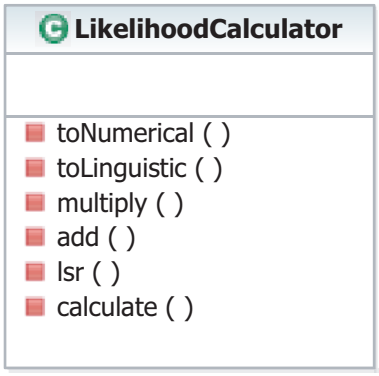

Figure A.17: The SCORE tool likelihood calculator

Appendix B

Software requirements specification

The following pages contain a full SRS for the SCORE tool. The document is prepared using a standard SINTEF template for the SINTEF ICT's hardware/software development process. The layout of the document thus deviates from the rest of this report, but on the other hand serves the purpose of being implemented as documentation for the SCORE tool within the organisation.

Requirement specification - the SCORE tool

SINTEF

www.sintef.no

SINTEF ICT

Trondheim
Address: N-7465 Trondheim
Location: O S Bragstads plass 2D
Telephone: +47 73 59 30 00
Fax: +47 73 59 43 99
……
Oslo
Address:… P.O. Box 124, Blindern
N-0314 Oslo
Telephone: +47 22 06 73 00
Fax: +47 22 06 73 50

Enterprise No.: NO 948 007 029 MVA

TITTEL

Requirement specification - the SCORE tool

AUTHOR(S)

Stig Torsbakken / Ketil Stølen

CLIENT(S)

SINTEF ICT

CLIENT'S REF.

CLASSIFICATION	FILE CODE	PROJECT NO.	NO. OF PAGES/APPENDICES
Open			9

ELECTRONIC CODE	PROJECT OWNER (NAME AND SIGNATURE)	QUALITY CONTROL OFFICER (NAME AND SIGNATURE)
requirements spesification.doc		

DATE APPROVED BY SINTEF ICT	**FOR SINTEF ICT (PROJECT MANAGER'S NAME AND SIGNATURE)**

DATE APPROVED BY CLIENT(S)	**FOR CLIENT(S) (NAME, POSITION, AND SIGNATURE)**

ABSTRACT

This report documents the requirements for the SCORE tool. The SCORE tool is a tool-supported method for risk estimation using CORAS diagrams and will be an integrated part of the CORAS tool. The requirements specification covers user requirements, system requirements and technical (non-functional) requirements.

Table of contents

1 Introduction 3
1.1 Scope/purpose of document 3
1.2 Overview of system 3
1.3 Document overview 3
1.4 Nomenclature 4

2 User requirements 5

3 System requirements 7
3.1.1 Functional requirements 7
3.1.2 Derived requirements 8

4 Technical requirements (non-functional requirements) 9

5 References 9

1 Introduction

This document describes top level requirements for a method to analyse qualitative information in the risk estimation phase during a risk analysis using the CORAS methodology. The work that this document deals with is a part of the master thesis with the title: “A tool-supported method for risk estimation using CORAS diagrams”[1].

1.1 Scope/purpose of document

This document establishes the functional and technical requirements for the development of the SCORE tool. As in most software development processes, the requirements specification contracts the deliverance from the developer(s) to the customer, but here the document also serves the purpose of success criteria upon which the master thesis will be evaluated.

The scope of the specification is limited to the technical design and implementation of the tool. An own section of the thesis is dedicated to the behavior of the tool’s underlying method for qualitative analysis/risk estimation.

1.2 Overview of system

The SCORE tool adds new functionality to the already existing system, *the CORAS tool*, enabling the risk analysts to handle qualitative information in a systematic manner. A thorough introduction to the tool and its methodology can be read in [1], but a brief introduction is given below.

The part of the system that the SCORE tool deals with is used during the risk estimation phase (activity 3 in the CORAS method [6]). During the related risk estimation meeting with the client, the risk analysis team collects qualitative data used to estimate the risk level of already identified risks. The values are measures for *likelihood* and *consequences* related to *treat scenarios*, *unwanted incidents* and *assets*. Much confusion and vague estimation has been proved through repeated field trials to come to pass, and the SCORE tool aims to assist the risk analysts to overcome the troublesome activity.

By breaking a problem down into smaller pieces we believe that humans are more capable to assign correct estimations to complex problems (divide and conquer). Where the CORAS methodology requires the analysts to come up with likelihood estimations for *unwanted incidents* which compound (possibly) of several paths leading to the incident, the SCORE tool allows to (only) estimate the level at the *threat scenarios* which may be by far less complex than the other. Then the SCORE tool applies the SCORE method to aggregate the estimation data granting the requested estimates for the unwanted incidents.

1.3 Document overview

Chapter 1 is an introduction to this paper that motivates the purpose and scope of the document and provides an overall overview of the system for which the requirements specification applies to. Next we give a brief, but more technical oriented introduction to the system in chapter 2. In chapter 3 we document the user requirements for the tool, while chapter 4 defines the system requirements in terms of functional and non-functional requirements.

1.4 Nomenclature

CORAS	
QAIC	A tool supported method for qualitative analysis in CORAS
SRS	Software Requirements Specification
TBC	To be completed
TBD	To be defined.
UI	Unwanted incident
TS	Threat scenario
A	Asset
Node	UI/TS

2 User requirements

The user requirements specify the external behavior of the system from a non-technical point of view. We present the requirements using textual notion, and they are the origin of the use cases presented in [1]. The user requirements shall embrace all functionality provided by the SCORE from a user's perspective.

[UR-1]: Select diagram element

- The user shall at any time be able to select any number of elements from the CORAS diagram and get the desired functionality described below ([UR-2], [UR-3] and [UR-4]).

Pre-conditions: CORAS diagram with diagram element(s) exists

Comment: This does not apply when the user is in the path view.

[UR-2]: Select single node

- The user shall at any time be able to select a single node from the CORAS diagram and get an overview of its properties its properties.

Pre-conditions: [UR-1], node exists

Comment: This does not apply when the user is in the path view.

[UR-3]: Select edge

- The user shall at any time be able to select an edge from the CORAS diagram and get an overview of its properties.

Pre-conditions: [UR-1], edge exists

Comment: This does not apply when the user is in the path view.

[UR-4]: Select multiple diagram elements

- The user shall at any time be able to select multiple nodes from the CORAS diagram and get the path view to handle a path.

Pre-conditions: [UR-1], nodes exists

[UR-5]: Assign likelihood value

- The user shall be able to assign a likelihood value to a node or initiate edge
- The user shall get legal likelihood values proposed, but also have the opportunity to manually define values.
- The user shall be able to choose between qualitative and quantitative likelihood values where available.

Pre-conditions: [UR-2] or [UR-3] conditionally

[UR-6]: Assign consequence value

- The user shall be able to assign a consequence value to a harm edge
- The user shall get legal consequence values proposed, but also have the opportunity to manually define values.
- The user shall be able to choose between qualitative and quantitative consequence values where available.

Pre-conditions: [UR-3] where edge is a harm edge

[UR-7]: Import likelihood values

- The user shall be able to import likelihood values from the likelihood tables in the CORAS tool (on-demand)
- The likelihood values shall be imported automatically when the diagram-editor is started from the CORAS tool

Pre-conditions: likelihood values are defined

[UR-8]: Import consequence values

- The user shall be able to import consequence values from the consequence tables in the CORAS tool
- The consequence values shall be imported automatically when the diagram-editor is started from the CORAS tool

Pre-conditions: consequence values are defined

3 System requirements

Below the functional requirements of the tool and how it shall be implemented with the CORAS tool are presented.

3.1.1 Functional requirements

The SCORE tool shall

[SR-1]: provide a pane for the various risk estimation functions containing:
- *property view* - for single node or edge
- *path view* - for multiple node
- *problem view* - to flag warnings
- *likelihood view* - to display likelihood table
- *consequence view* - to display consequence tables (one for each asset)

[SR-2]: display properties in the property view when a single node is selected
Properties:
- *type* - the element type
- *body* - content of the element
- *likelihood* - the element's assigned likelihood value
- *consequence* - the element's assigned consequence value

[SR-3]: display properties in the property view when an edge is selected
Properties:
- *type* - the edge type
- *likelihood/consequence* - the edge's assigned likelihood or consequence value
- *source* - the initiating element
- *target* - the initiated element

[SR-4]: display selected path in the path view when multiple nodes are selected
- perform consistency check on selected path
- flag warning if path is inconsistent

[SR-5]: allow for likelihood values to be assigned to a node or edge when selected
- propose legal values
- allow for manually selected values
- pin inconsistency warning if manually selected values are assigned
- present the legal values through a drop-down box

[SR-6]: allow for consequence values to be assigned to a harm edge when selected
- propose legal values
- allow for manually selected values
- pin inconsistency warning if manually selected values are assigned
- present the legal values through a drop-down box

[SR-7]: automatically calculate likelihood on assignment or changes in likelihood values
- perform a *least set required (LSR)* check to determine if the affected diagram elements holds the required likelihood values for calculation
- calculate the likelihood values according to the SCORE method upon validation of the LSR-check
- pin warning if LSR-check fails

3.1.2 Derived requirements

The system requirements above are mostly general requirements that need to be broken down into further detail. Below we present their derived requirements annotated with [DR-n,m], where n is the SR number to which it belongs and m is the derived number.

[DR-1.1]: the property view shall automatically be updated to hold the properties of the selected element immediately when a new element gains focus
- when no element is selected, the property view shall display the properties of the diagram

[DR-1.1]: the property view shall be composed of a two-column table where each row is assigned to an unique property
- column one describes which property
- the second column is an interactive field for the user to alter the properties

Comment: Some fields of the interactive column may be fixed, such as "Type", since they are not changeable

[DR-1.1]: the path view shall automatically be updated immediately when a new path gains focus

[DR-1.2]: the likelihood view shall be composed of a two-column table of five rows that defines the linguistic variables and their corresponding probabilistic intervals (numerical)

[DR-1.3]: the consequence view shall be composed of a two-column table of five rows that defines the linguistic variables and their corresponding numerical intervals

[DR-2.1]: the *type* row shall be fixed and displaying the element's type (i.e.: threat scenario, unwanted incident)

[DR-2.2]: the *body* row shall be interactive
- the row shall allow the user to alter the body content
- the row shall at any time display the actual and updated content of the node

[DR-2.3]: the *likelihood* row shall be interactive
- the row shall allow the user to alter the likelihood value through a drop-down box containing legal likelihood values or manually insert an inconsistent value
- the row shall at any time display the actual and updated likelihood value of the node

[DR-2.4]: the *consequence* row shall be inactive
- consequence values shall only be allowed to insert on harm edges

[DR-3.1]: the *type* row shall be fixed and displaying the edge's type (i.e.: initiate edge, harm edge)

[DR-3.2]: the *body* row shall be inactive
- edges shall not have any body other than likelihood/consequence values

[DR-3.3]: the *likelihood/consequence* row shall be interactive
- the row shall allow the user to alter the likelihood value through a drop-down box containing legal likelihood values or manually insert an inconsistent value if the edge is an initiate edge
- the row shall allow the user to alter the consequence value through a drop-down box containing legal likelihood values or manually insert an inconsistent value if the edge is a harm edge
- the row shall at any time display the actual and updated likelihood or consequence value of the edge

[DR-3.4]: the *source* row shall be fixed

- the row shall display the edge's source (i.e.: threat scenario, unwanted incident)

[DR-3.5]: the *target* row shall be fixed

- the row shall display the edge's target (i.e.: threat scenario, unwanted incident, asset)

4 Technical requirements (non-functional requirements)

[TR-1]: The SCORE tool shall be implemented with purpose and possibility of being distributed with the future release version 3.0 of the CORAS tool

[TR-2]: The SCORE shall be implemented with low coupling internally such that each internal component is easy to alter separately

[TR-3]: The SCORE tool shall allow the standalone editor of the CORAS risk modeling language to run independent of the CORAS tool (full version)

- The editor shall provide the ability to manually insert likelihood values when the likelihood table is not present (refer to [SR-5]:)
- The editor shall provide the ability to manually insert consequence values when the consequence tables are not present (refer to [SR-6]:)

5 References

[1] Torsbakken, S.: *A tool-supported method for risk estimation using CORAS diagrams.* Unpublished.
[2] Hogganvik, I. and Stølen, K.: *A Graphical Approach to Risk Identification, Motivated by Empirical Investigations*.
[3] Hogganvik, I. and Stølen, K.: *Specification of the language including modeling guidelines*
[4] The CORAS project homepage: http://coras.sourceforge.net/
[5] The CORAS/Sourceforge home site: http://sourceforge.net/projects/coras
[6] Folker den Braber, Gyrd Brændeland, Heidi E. I. Dahl, Iselin Engan, Ida Hogganvik, Mass S. Lund, Bjørnar Solhaug, Ketil Stølen, Fredrik Vraalsen: *The CORAS Model-based Method for Security Risk Analysis.* Unpublished.
[7] Brændeland, G and Hogganvik, I and Engan, I: *Evaluation of the method and tool used during the 7th field trial of SECURIS*

Appendix C

Security risk analysis of Stubrud sjakk-shop

The following pages contain the CORAS security risk analysis of Stubrud sjakk-shop. The document is prepared using the CORAS tool, thus the layout of the document deviates from the rest of this report.

SINTEF

SINTEF ICT

Address: P.O.Box 124, Blindern
0314 Oslo NORWAY
Location: Forskningsveien 1
0373 Oslo
Telephone: +47 22 06 73 00
Fax: +47 22 06 73 50

Enterprise No.: NO 948 007 029 MVA

SINTEF REPORT

TITLE

CORAS security risk analysis of Stubrud sjakk-shop

AUTHOR(S)

Stig Torsbakken / Ketil Stølen

CLIENT(S)

Stubrud sjakk-shop

REPORT NO.	CLASSIFICATION	CLIENTS REF.	
	Open	Jan Magne Stubrud	
CLASS. THIS PAGE	ISBN	PROJECT NO.	NO. OF PAGES/APPENDICES
Open			32/1
ELECTRONIC FILE CODE		PROJECT MANAGER (NAME, SIGN.)	CHECKED BY (NAME, SIGN.)
Document2			
FILE CODE	DATE	APPROVED BY (NAME, POSITION, SIGN.)	
	2007-04-29		

ABSTRACT

This report presents the results from a security risk analysis performed at *Stubrud sjakk-shop*. The target of the analysis is the company's e-commerce IT infrastructure. In particular; we focus on availability of the local computer system and confidentiality of sensitive data.

The security risk analysis is performed as a case study to evaluate the SCORE project and therefore concentrate with special emphasis on the risk estimation phase.

KEYWORDS	ENGLISH	NORWEGIAN
GROUP 1	ICT	IKT
GROUP 2	Information systems	Informasjonssystemer
SELECTED BY AUTHOR	Model-based risk analysis	Modellbasert risikoanalyse

1. Introduction

This report presents the results from a security risk analysis performed at STUBRUD SJAKK SHOP. The target of the analysis is the company's e-commerce IT infrastructure. The security risk analysis is performed as a case study to evaluate the SCORE project and therefore concentrate with special emphasis on the risk estiamtion phase, but is at the same time commissioned by STUBRUD SJAKK SHOP (hereby refered to as Sjakk shop) as a commercial security risk analysis.

1.1 Objective

The security risk analysis (hereby refered to as the risk analyisis or RA) has two main objectives to accommodate the interests of both the Sjakk shop and the SCORE evaluation. First of all we identify and analyse risks related to the target of analysis in the Sjakk shop. Main focus for the target of analysis is the availability of the local computer system and confidentiality of sensitive data. The SCORE project's main objective is to evaluate the SCORE method and tool which concerns the phase of risk estimation.

1.2 Scope

The risk analysis is restricted to handle only the Sjakk shop processes in which affects the IT infrastructure of the business. It also focuses on the system as it works today and the system parts that are under daily maintenance of the Sjakk shop manager. This means that physical assets like product stock and the outsourced webhosting of the business' home site and accounting system is out of scope. However; we include the possibilty of future expansion of the present e-commerce to become a full e-trade site with product payment as an online service, and the availability of the home site as to what lies under the Sjakk shop manager's domain of influence.

1.3 Team and plan

The persons involved in the risk analysis:

Name	Role	Organisation	Background/expertise
Stig Torsbakken	RA leader	SINTEF	Risk analysis
Jan Magne Stubrud	Target owner/field expert	Sjakk shop	Business manager

Table: 1

The risk analysis is conducted in accordance with the CORAS method for Model-based Risk Analysis developed in the CORAS and SECURIS projects. The CORAS risk management process consists of the following five sub-processes:

- *Context identification*: Identify the context of the analysis. Describe the application(s) and/or organization, identify usage scenarios, the assets of the target of analysis, and the risk evaluation criteria.
- *Risk identification*: Identify the potential threats to assets and the vulnerabilities of these assets. Identify unwanted incidents.
- *Risk estimation*: Estimate the consequence and frequency value of each unwanted incident identified in the risk identification phase.

- *Risk evaluation*: Identify the level of risk associated with the unwanted incidents already identified and assessed in the previous sub-processes.
- *Risk treatment*: Address the treatment of the identified risks, and how to prevent the unacceptable risks.

1.4 Terminology

- Analysis leader: The person who leads the structured brainstorming sessions and maintains contact with the client between sessions.
- Analysis secretary: The person who documents the results during the sessions with the client.
- Analysis team: The group of persons who conduct the risk analysis, including the analysis leader and analysis secretary.
- Risk analysis: The process of context identification, risk identification, risk estimation, risk evaluation and risk treatment.
- Structured workshop: A workshop involving experts on the target of analysis to identify and analyse risks.
- Target of analysis: The organization or system that is the object of analysis. External entities that may affect the security of the target are also analysed.
- Asset: Anything considered to be of value for the client; thus something the client wants to protect
- Vulnerability: A weakness or flaw of the system that a threat may take advantage of
- Threat: A human or a non-human entity that may cause an event that harms an asset (deliberatly or accidentally)

The figure below explains the graphical notion used throughout the risk analysis:

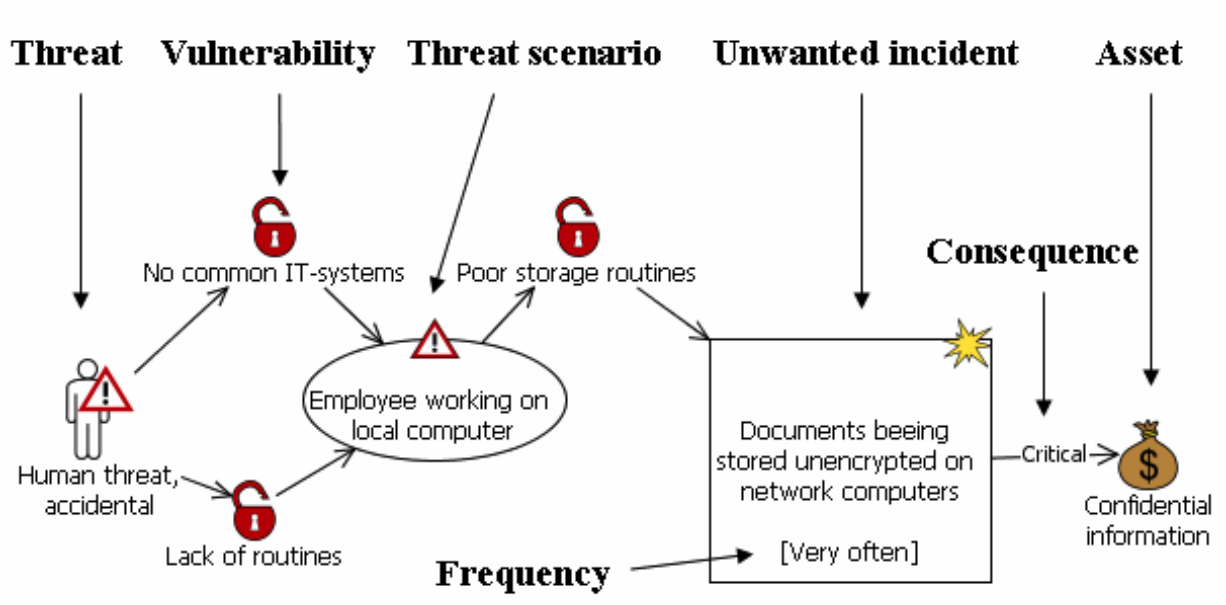

Figure: 1

2. Context identification

The purpose of the context identification is to characterise the target of the analysis and its environment and to agree on fundamental assumptions. Most of the documentation on the target of analysis is located in appendix 1. However, in this section we include a short description of the business and the persons involved in the risk analysis as the target of evaluation. Thereafter we present the assets and show an overall picture of the system. At last we define the likelihood and consequence scales and the risk evaluation matrices.

What is left out of the main report and included in appendix 1, is the formal and detailed system specifications and the system activities descriptions including activity diagrams and sequence diagrams of the system processes.

2.1 Target of evaluation

The target of evaluation is the IT-infrastructure of a one man enterprise owned and managed by Jan Magne Stubrud. The business Sjakk shop is an e-commerce supplier of chess products for the general public, schools and chess clubs. Stubrud runs his business from his private home and has the full responsibility and does all work tasks himself without employees.

The sjakk-shop holds a good reputation among chess interested in Norway as a reliable supplier of chess products. The corporate policy of sjakk-shop is summarised to:

The sjakk-shop shall

- only have satisfied customers
- deliver on time
- always be available
- provide good information to the customers throughout the purchasing process

The sjakk-shop is mainly located at three physical locations: the company management center including a local computer system and a product stock at Stubrud's private home, the web site and web management at a third party web hosting company, and the accounting system at a third party accountancy firm. Figure 2 explains the situation visually:

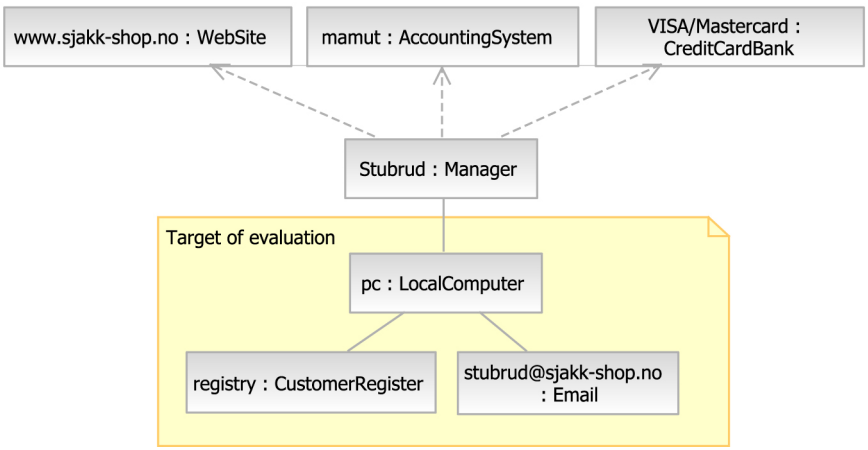

Figure: 2

The outsourced third party targets fall outside the scope of the risk analysis since the security management of their systems are not within the field of responsibility of the sjakk-shop manager. The communication between the sjakk shop, its customers and suppliers is mainly through internet, e-mail or telephone. An example of a typical ordering process is sketched in figure 5 under the system description.

2.2 Asset identification

In figure 3 you can see a full overview of all the identified assets with respect to the TOE:

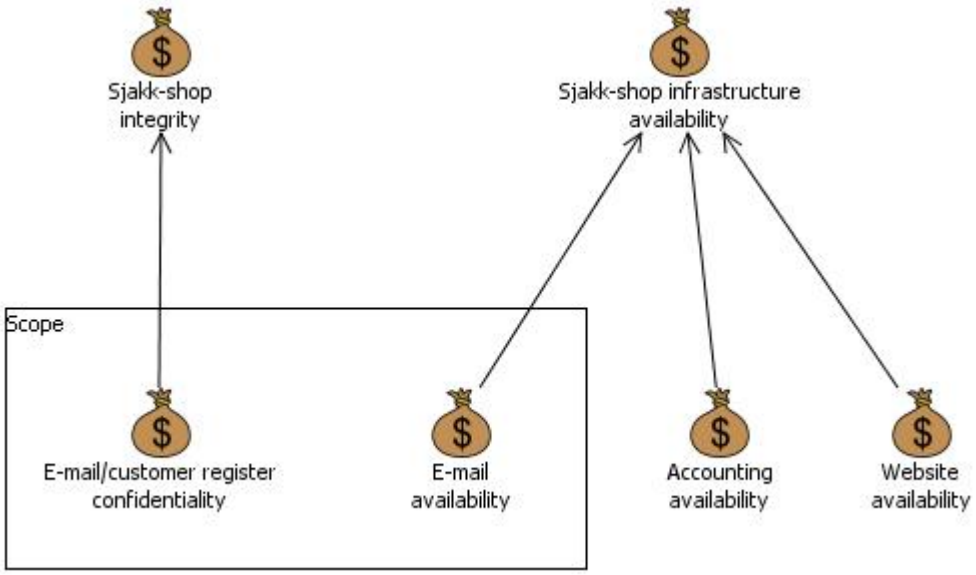

Figure: 3

In order to keep the risk analysis limited to a reasonable size, we choose to focus exclusively on direct assets and only those of highest importance and relevance. Table 2 lists the selected assets with their respective estimation of value.

Asset ID	Description	Value
E-mail/customer register confidentiality	The confidentiality of the company's e-mail and customer registry located at the local computer system	375 000
E-mail availability	The availability of the the sjakk-shop e-mail	25 000

Table: 2

2.3 System description

To illustrate most central business activities, we here present an overall use case diagram. Note that the diagram only contains the use cases within scope.

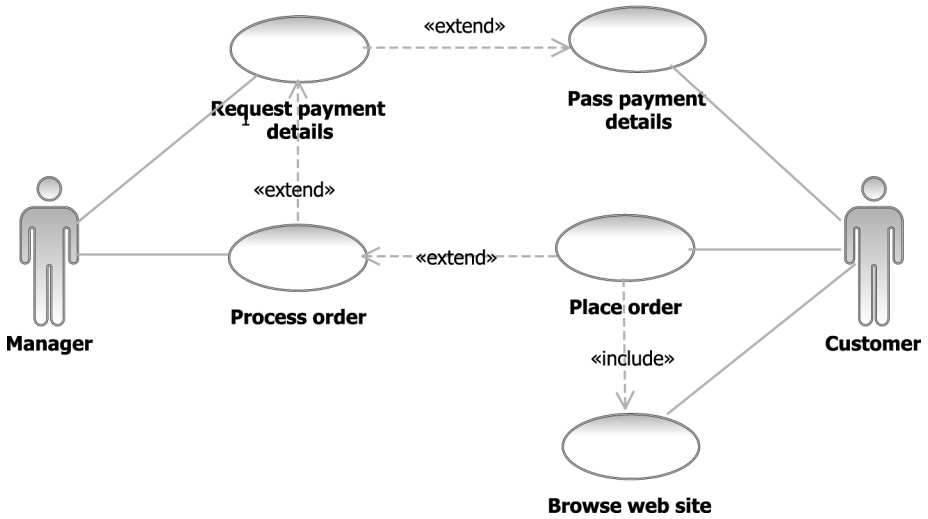

Figure: 4

From the use cases you see that they are in general concerned around the product ordering acitivities. What happens when a customer orders a product is best described with the sequence diagram below:

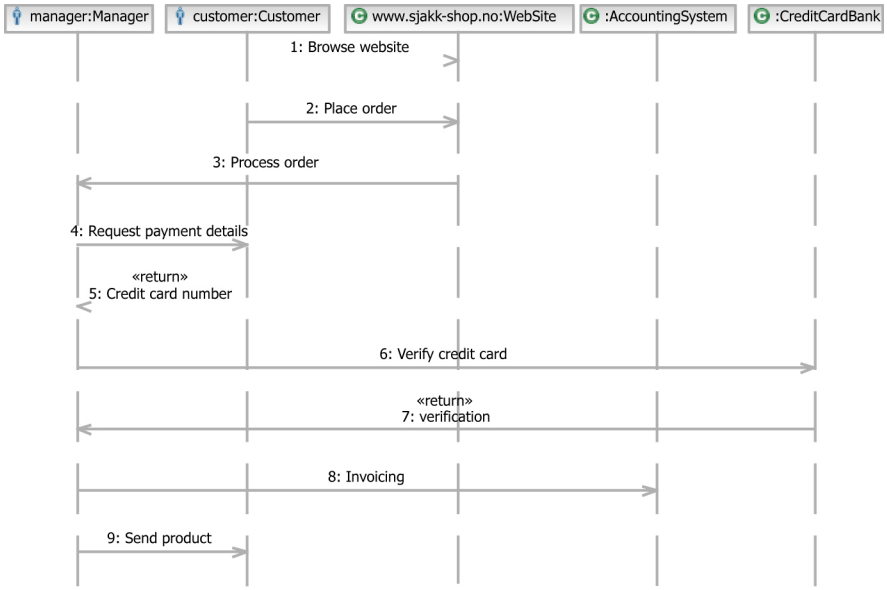

Figure: 5

In words; when a customer has placed an order at the sjakk-shop website, the manager receives this as an e-mail (on his local computer normally, but not necessarily) with the order details. He then contacts the customer on telephone (preferably, but e-mail correspondance could occur) and requests payment details (credit card number or invoice details) that in turn is validated with the credit card company. Finally, he does requisite invoicing at his accounting system and ships the ordered product.

2.4 Frequency and consequence scales

When we evaluate which risks we can accept and which we have to treat, we assign the risks likelihood and consequence values to be able to compare them. Below we present the likelihood and consequence values scales used for this purpose.

Frequency values (Normalised values pr. year in parenthesis)	
Almost never	Yearly (<1)
Very seldom	Monthly (1 - 12)
Seldom	Weekly (13 – 122)
Often	Every 1-3 days (123 – 365)
Very often	Daily (>365)

Figure: 6

Likelihood can be represented either as frequency values above that indicates a statistical measure based on known history or estimated data, or as probability values as given below in figure 7. The CORAS method does not define both representations, but because of the SCORE method we define both here, event though their semantics may overlap.

Probability values	
Very low	[0 - 0.1]
Low	<0.1 - 0.3]
Medium	<0.3 - 0.6]
High	<0.6 - 0.8]
Very high	<0.8 - 1]

Figure: 7

Consequence values Loss of income (1000 NOK) (E-mail/customer register confidentiality)	
Insignificant	0 – 10
Moderate	11 – 25
Large	26 – 50
Critical	51 – 150
Catastrophic	>150

Figure: 8

Consequence values Inaccessible time (E-mail availability)	
Insignificant	0 to 3 hours
Moderate	4 hours to 1 day
Large	2 to 5 days
Critical	6 to 14 days
Catastrophic	>14 days

Figure: 9

2.5 Risk evaluation matrices

To compare the identified risks and establish the level of risk that Sjakk shop can accept and what level that needs to be evaluated for treatment, we use the risk evaluation matrices below to define the risk evaluation criteria. If a risk ends up in the green area, it is accepted, but otherwise we have to look into the risk for possible treatment since it falls outside the accepted area.

Loss of income (E-mail/customer register confidentiality)					
Consequence	**Frequency**				
	Almost never	Very seldom	Seldom	Often	Very often
Insignificant					
Moderate					
Large					
Critical					
Catastrophic					

Figure: 10

Inaccessible time (E-mail availability)					
Consequence	**Frequency**				
	Almost never	Very seldom	Seldom	Often	Very often
Insignificant					
Moderate					
Large					
Critical					
Catastrophic					

Figure: 11

3. Risk identification and estimation

The risk identification section presents the identified risks.

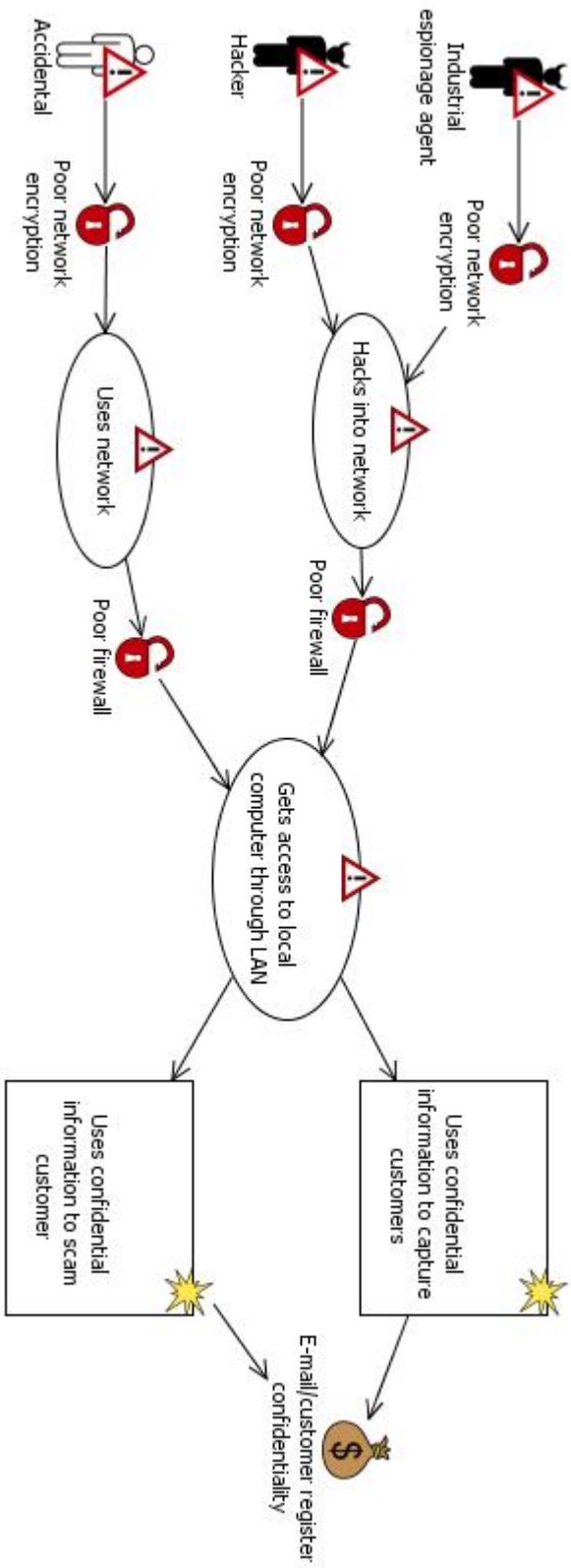

Figure: 12

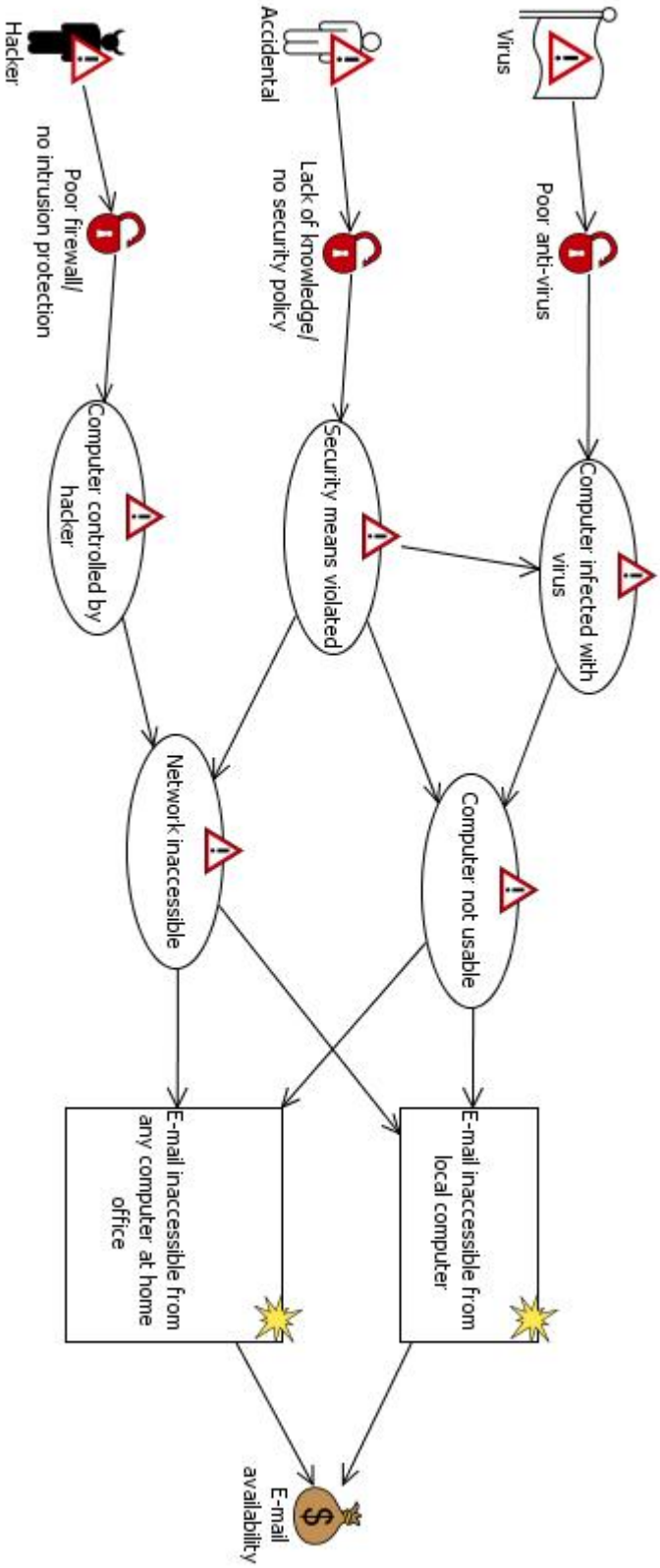

Figure: 13

Same diagrams with risk estimations:

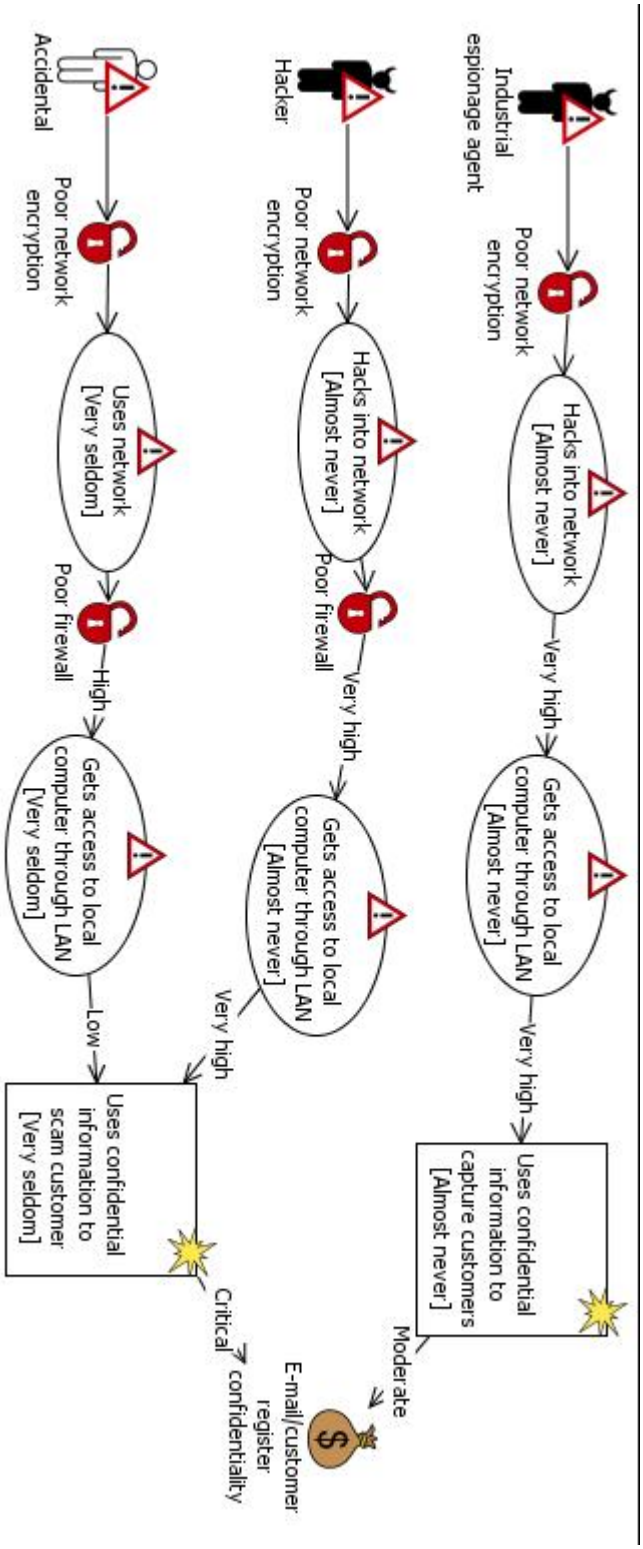
Industrial espionage agent
Poor network encryption
Hacks into network [Almost never]
Very high
Gets access to local computer through LAN [Almost never]
Very high
Uses confidential information to capture customers [Almost never]
Moderate
E-mail/customer register
Hacker
Poor network encryption
Hacks into network [Almost never]
Poor firewall
Very high
Gets access to local computer through LAN [Almost never]
Very high
Critical
confidentiality
Accidental
Poor network encryption
Uses network [Very seldom]
Poor firewall
High
Gets access to local computer through LAN [Very seldom]
Low
Uses confidential information to scam customer [Very seldom]

Figure: 14

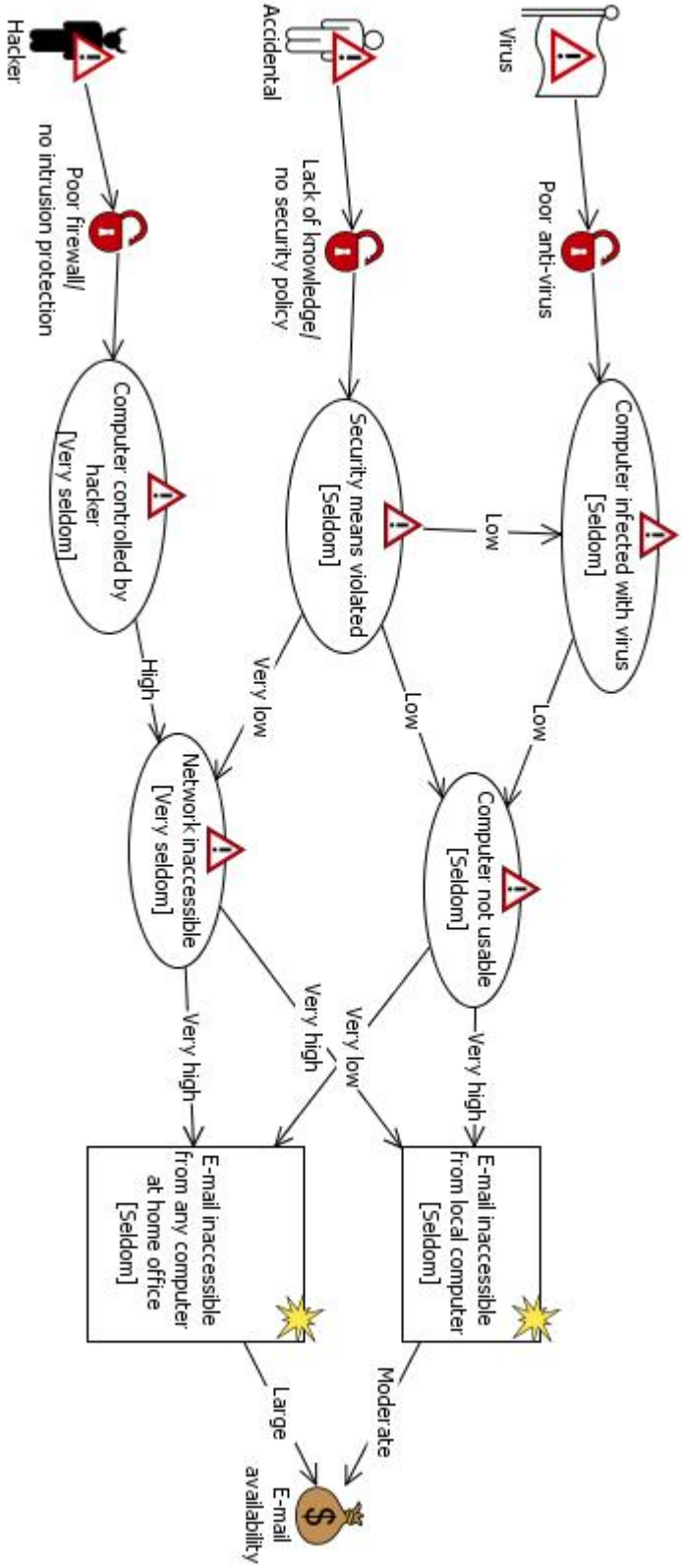
Virus
Accidental
Hacker
Poor anti-virus
Lack of knowledge/
no security policy
Poor firewall/
no intrusion protection
Computer infected with virus
[Seldom]
Security means violated
[Seldom]
Computer controlled by
hacker
[Very seldom]
Low
Low
High
Very low
Low
Network inaccessible
[Very seldom]
Computer not usable
[Seldom]
Very high
Very low
Very high
Very high
E-mail inaccessible
from any computer
at home office
[Seldom]
E-mail inaccessible
from local computer
[Seldom]
Large
Moderate
E-mail
availability

Figure: 15

4. Risk evaluation

4.1 Risk diagrams

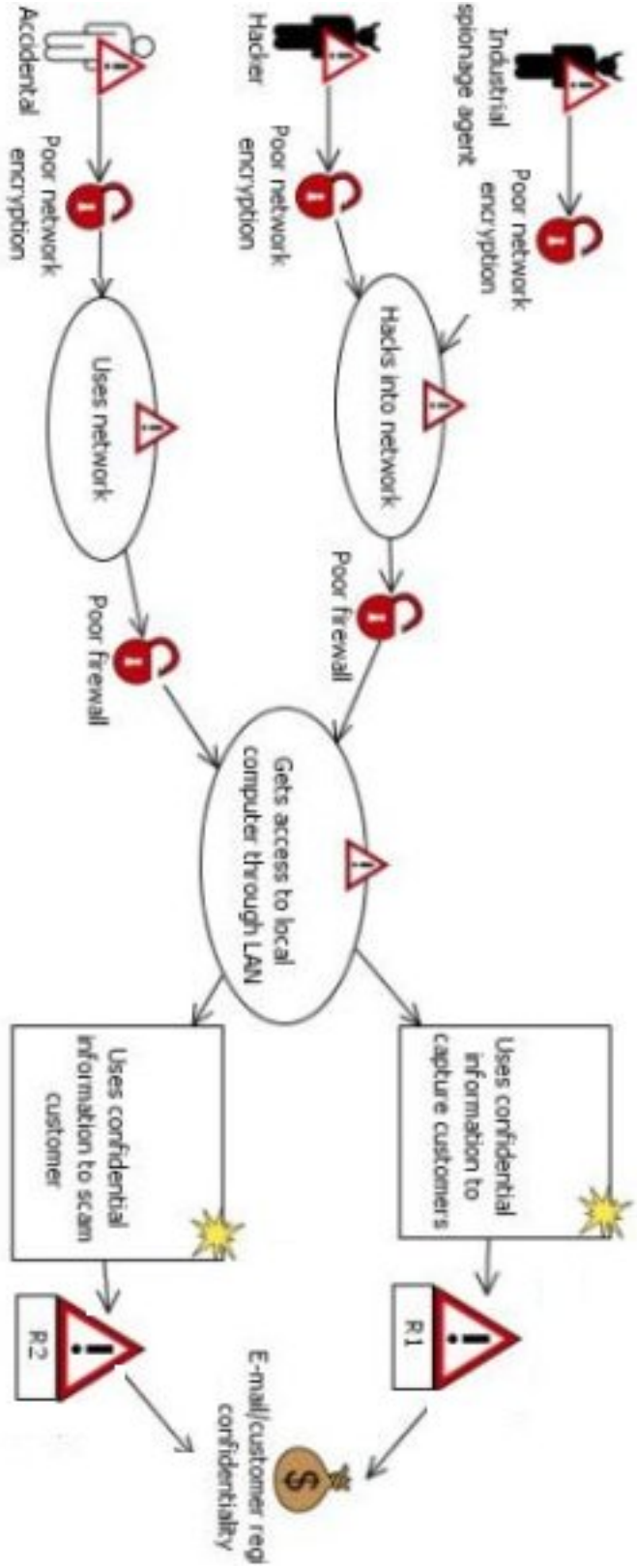

Figure: 16

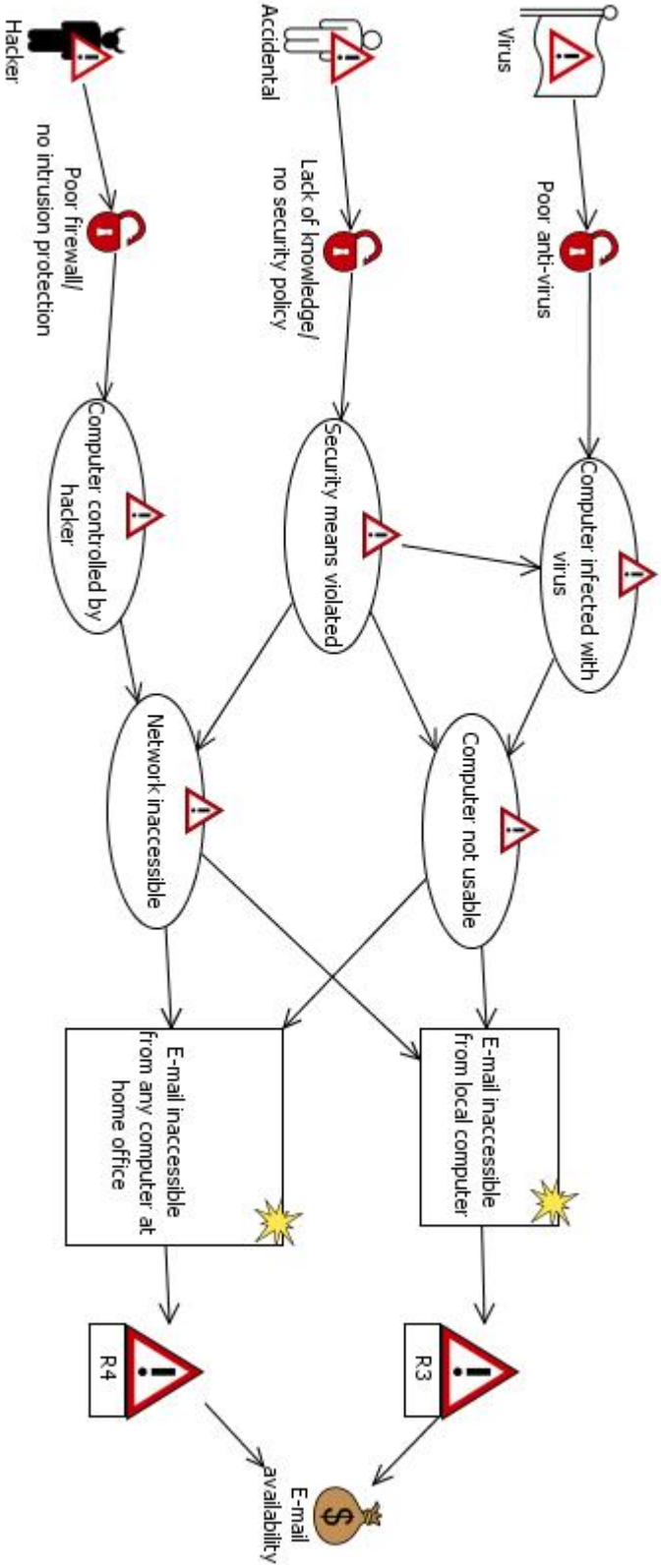
Virus
Accidental
Hacker
Poor anti-virus
Lack of knowledge/
no security policy
Poor firewall/
no intrusion protection
Computer infected with
virus
Security means violated
Computer controlled by
hacker
Computer not usable
Network inaccessible
E-mail inaccessible
from local computer
E-mail inaccessible
from any computer at
home office
R3
R4
E-mail
availability

Figure: 17

4.2 Risk table

Risk ID	Asset ID	Threat	Incident	Consequence Value	Frequency Value
R1	E-mail/customer register confidentiality	Industrial espionage agent	Uses confidential information to capture customers	Moderate	Almost never
R2	E-mail/customer register confidentiality	Hacker, Accidental threat	Uses confidential information to scam customer	Critical	Very seldom
R3	E-mail availability	Virus, Accidental threat, Hacker	E-mail inaccessible from local computer	Moderate	Seldom
R4	E-mail availability	Virus, Accidental threat, Hacker	E-mail inaccessible from any computer at home office	Large	Seldom

Table: 3

4.3 Risk evaluation matrices with risks

The risk evaluation matrices illustrates the risk level for each risk graphically. We easily get an overview of what risks we have to evaluate for treatment.

Loss of income (E-mail/customer register confidentiality)					
Consequence	**Frequency**				
	Almost never	Very seldom	Seldom	Often	Very often
Insignificant					
Moderate	**R1**				
Large					
Critical	**R2**				
Catastrophic					

Figure: 18

Inaccessible time (E-mail availability)					
Consequence	**Frequency**				
	Almost never	Very seldom	Seldom	Often	Very often
Insignificant					
Moderate			**R3**		
Large					
Critical			**R4**		
Catastrophic					

Figure: 19

We see that R1 is accepted, while **R2**, **R3** and **R4** needs to be treated.

5. Risk treatment

The following risk diagrams suggests action to take towards the risks R2, R3 and R4:

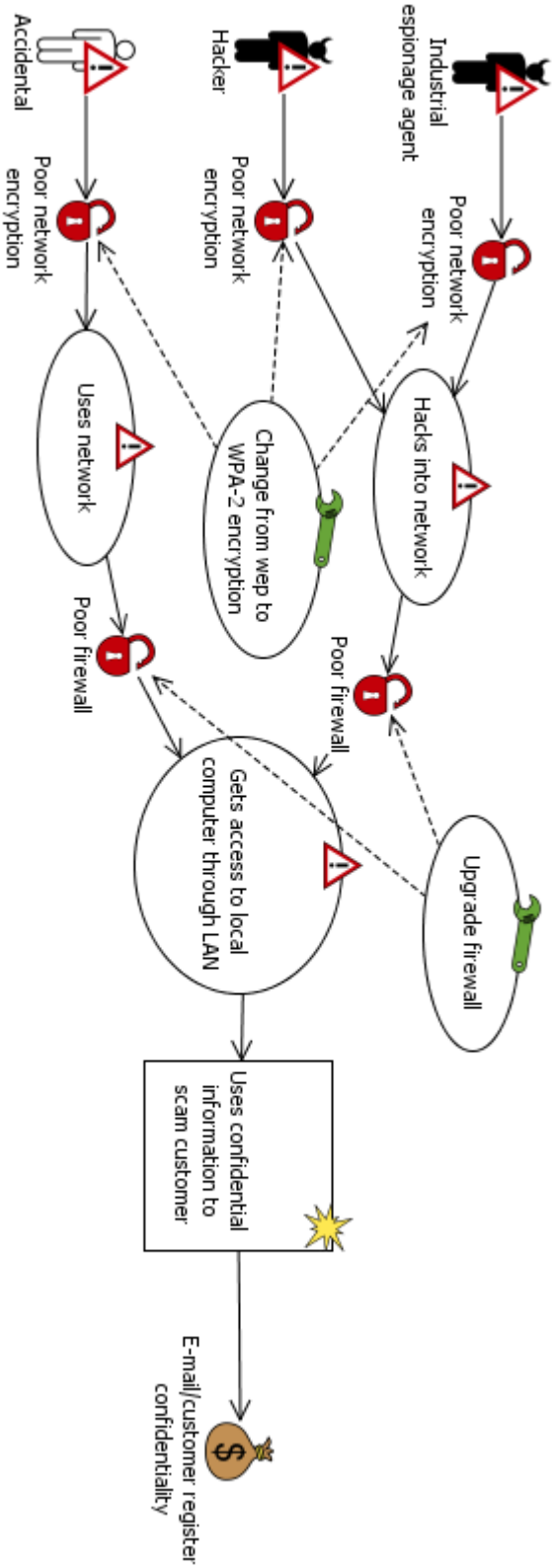

Figure: 20

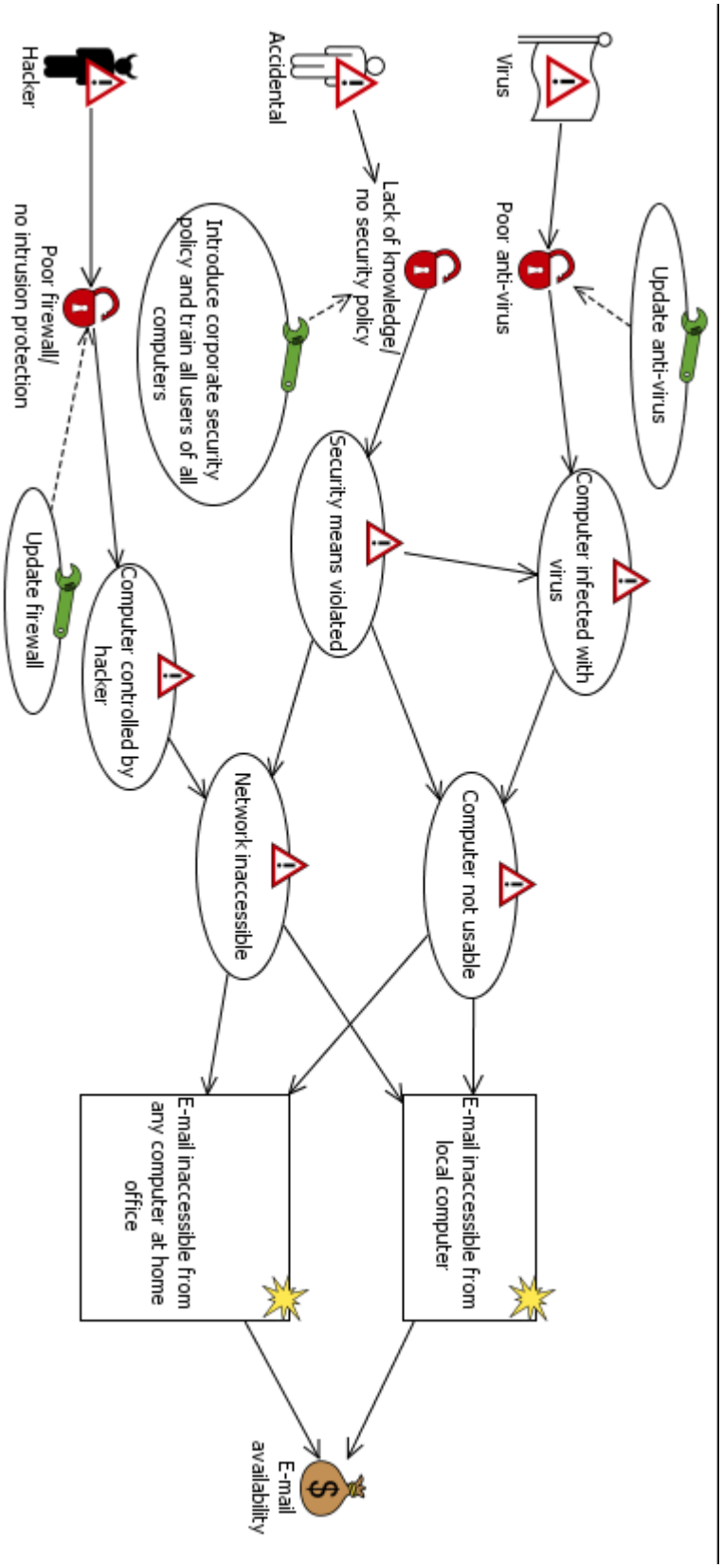
Virus
Accidental
Hacker
Poor anti-virus
Lack of knowledge/
no security policy
Poor firewall/
no intrusion protection
Update anti-virus
Introduce corporate security
policy and train all users of all
computers
Update firewall
Security means violated
Computer infected with
virus
Computer controlled by
hacker
Network inaccessible
Computer not usable
E-mail inaccessible from
any computer at home
office
E-mail inaccessible from
local computer
E-mail
availability

Figure: 21

6. Conclusions and recommendations

This report documents the results from the risk analysis performed at Sturbrud sjakk-shop of the company's IT-infrastructure. Main focus of the risk anaysis was in particular e-mail and customer registrer analysed with respect to confidentiality and availability.

Assets are defined as: *e-mail/customer register confidentiality* and *e-mail availability* in the context identification phase. During this phase there were also defined frequency, probability and consequence values together with risk evaluation critera upon which the risks are evaluated for acceptance or treatment.

To the respective assets, the following threats were found:

E-mail/customer register confidentiality:

- Industrial espionage agent uses confidential information to capture customers
- Hacker or accidental uses confidential information to scam customers

E-mail availability:

- Virus, accidental or hacker makes e-mail inaccessible from local computer
- Virus, accidental or hacker makes e-mail inaccessible from any computer at home office

To the unwanted incidents identified for each of the scenarios above, there were assigned likelihood and consequence values that together with their respective asset constitutes the risk to each of the assets. Based on the risk evaluation criteria, **R1** to the asset *e-mail/customer register confidentiality* was eliminated, while **R2**, **R3** and **R4** needed to be evaluated for treatment.

The suggested treatments generally involves keeping firewall and anti-virus up to date and encrypt wireless LAN with WPA-2 instead of WEP. Furthermore, we recommend to introduce a corporate security policy. All users of the company's IT-infrastructure need to be made aware of such policy and suffiniently trained.

7. Appendix 1: Context Identification documentation

7.1 System specifications

Here we present the formal and detailed system specifications.

The total overview

Below we show an informal overview of the business followed by a UML class diagram that tries to formally specify the entities from the first figure and assigning them appropriate attributes and methods.

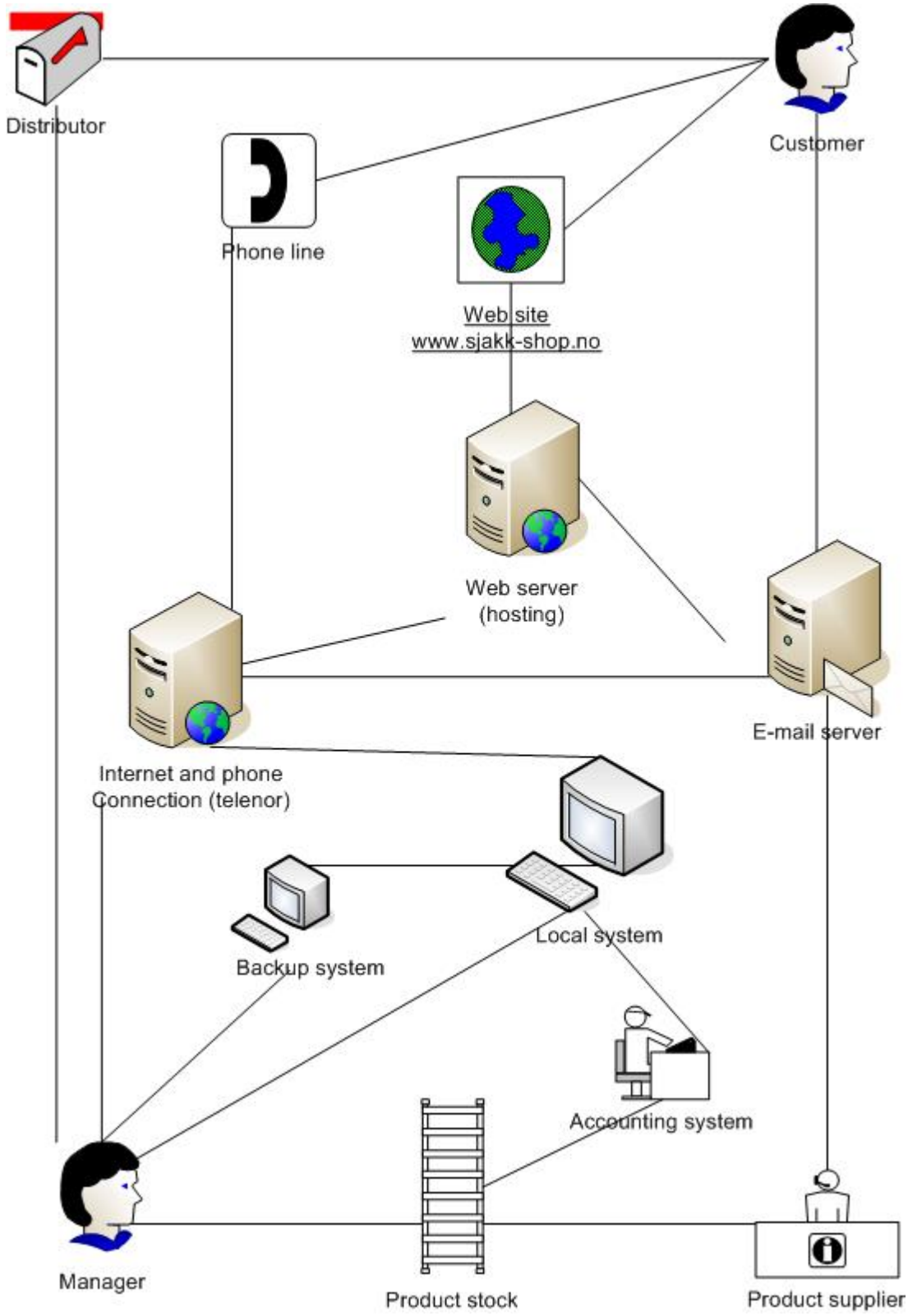

Figure: 22

And the UML specifications:

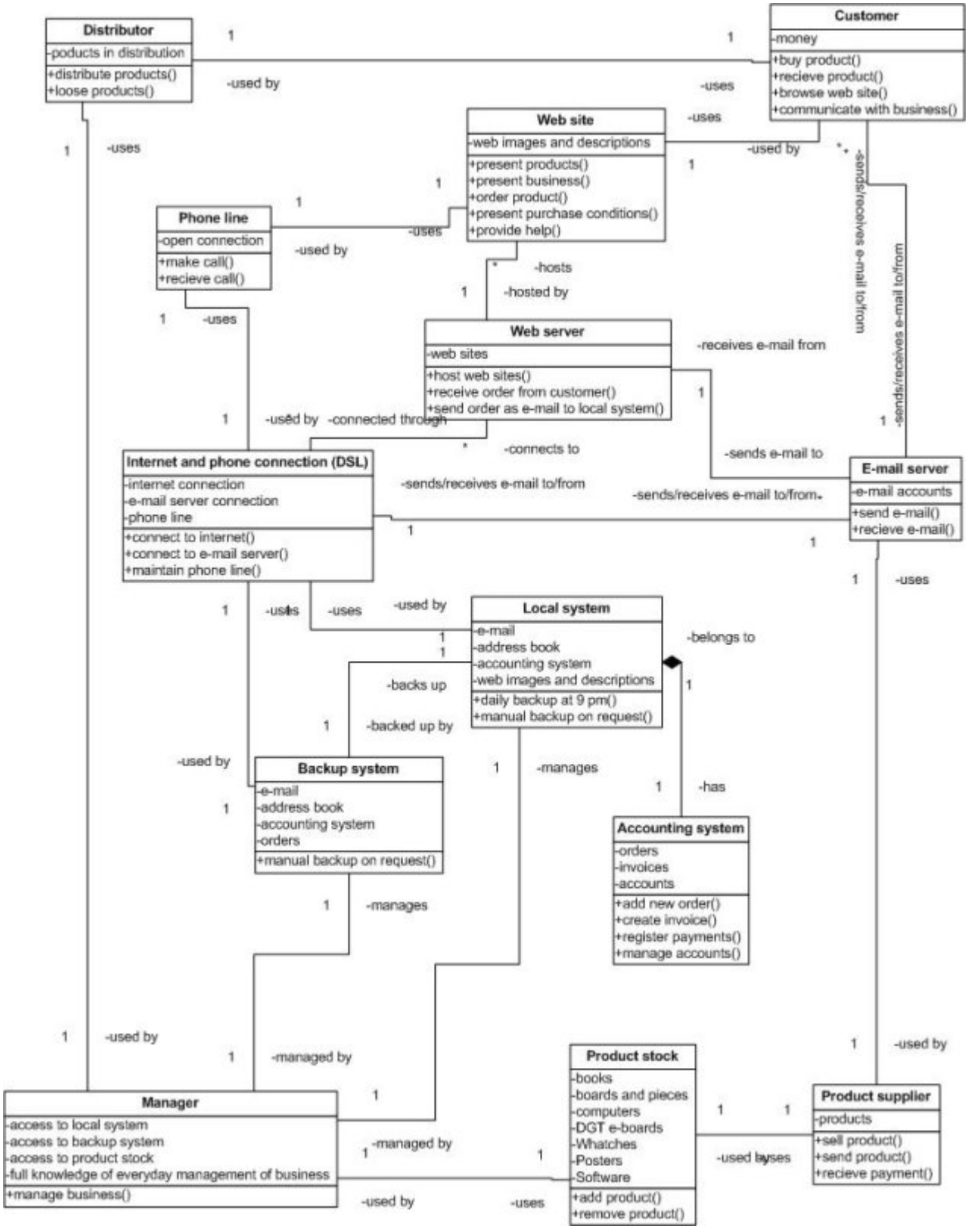
Distributor
-poducts in distribution
+distribute products()
+loose products()
Customer
-money
+buy product()
+recieve product()
+browse web site()
+communicate with business()
Web site
-web images and descriptions
+present products()
+present business()
+order product()
+present purchase conditions()
+provide help()
Phone line
-open connection
+make call()
+recieve call()
Web server
-web sites
+host web sites()
+receive order from customer()
+send order as e-mail to local system()
Internet and phone connection (DSL)
-internet connection
-e-mail server connection
-phone line
+connect to internet()
+connect to e-mail server()
+maintain phone line()
E-mail server
-e-mail accounts
+send e-mail()
+recieve e-mail()
Local system
-e-mail
-address book
-accounting system
-web images and descriptions
+daily backup at 9 pm()
+manual backup on request()
Backup system
-e-mail
-address book
-accounting system
-orders
+manual backup on request()
Accounting system
-orders
-invoices
-accounts
+add new order()
+create invoice()
+register payments()
+manage accounts()
Manager
-access to local system
-access to backup system
-access to product stock
-full knowledge of everyday management of business
+manage business()
Product stock
-books
-boards and pieces
-computers
-DGT e-boards
-Whatches
-Posters
-Software
+add product()
+remove product()
Product supplier
-products
+sell product()
+send product()
+recieve payment()
-used by
-uses
-hosts
-hosted by
-receives e-mail from
-sends/receives e-mail to/from
-connected through
-connects to
-sends e-mail to
-belongs to
-backs up
-backed up by
-manages
-has
-managed by
-used buses

Figure: 23

7.2 System activities

Activity diagrams

Activity diagram for the use cases *order product* and *process order*:

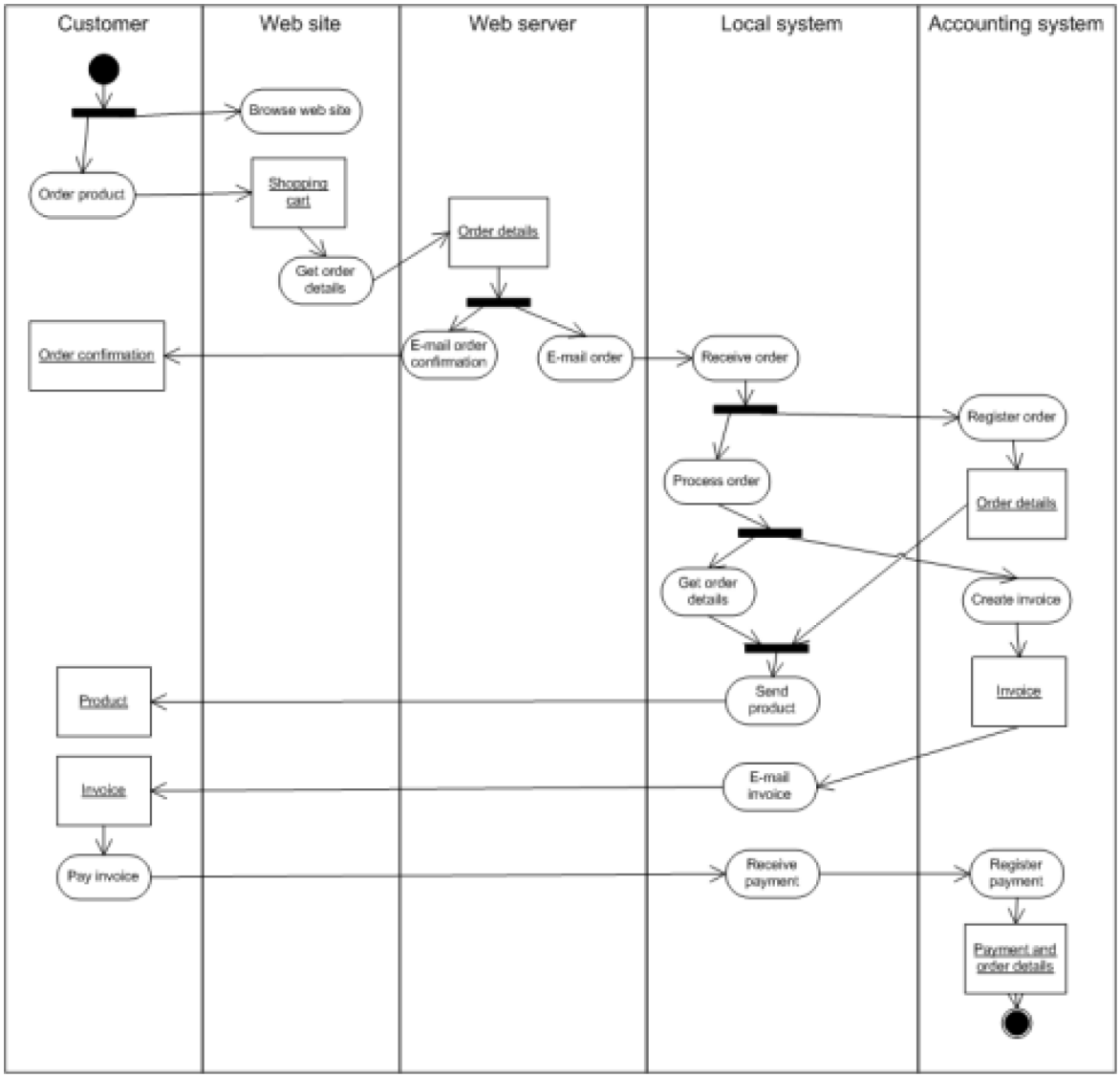

Figure: 24

Sequence diagrams

Order product:

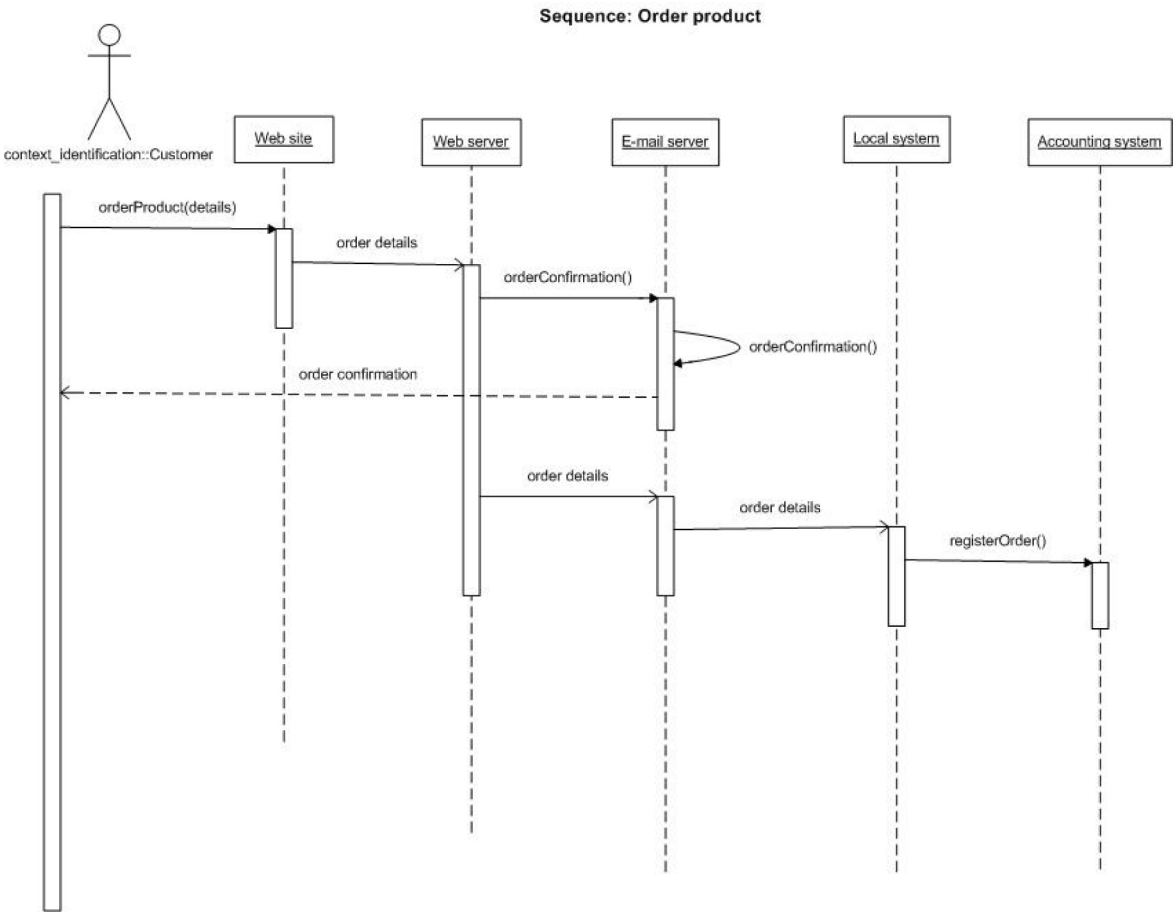

Figure: 25

Process order:

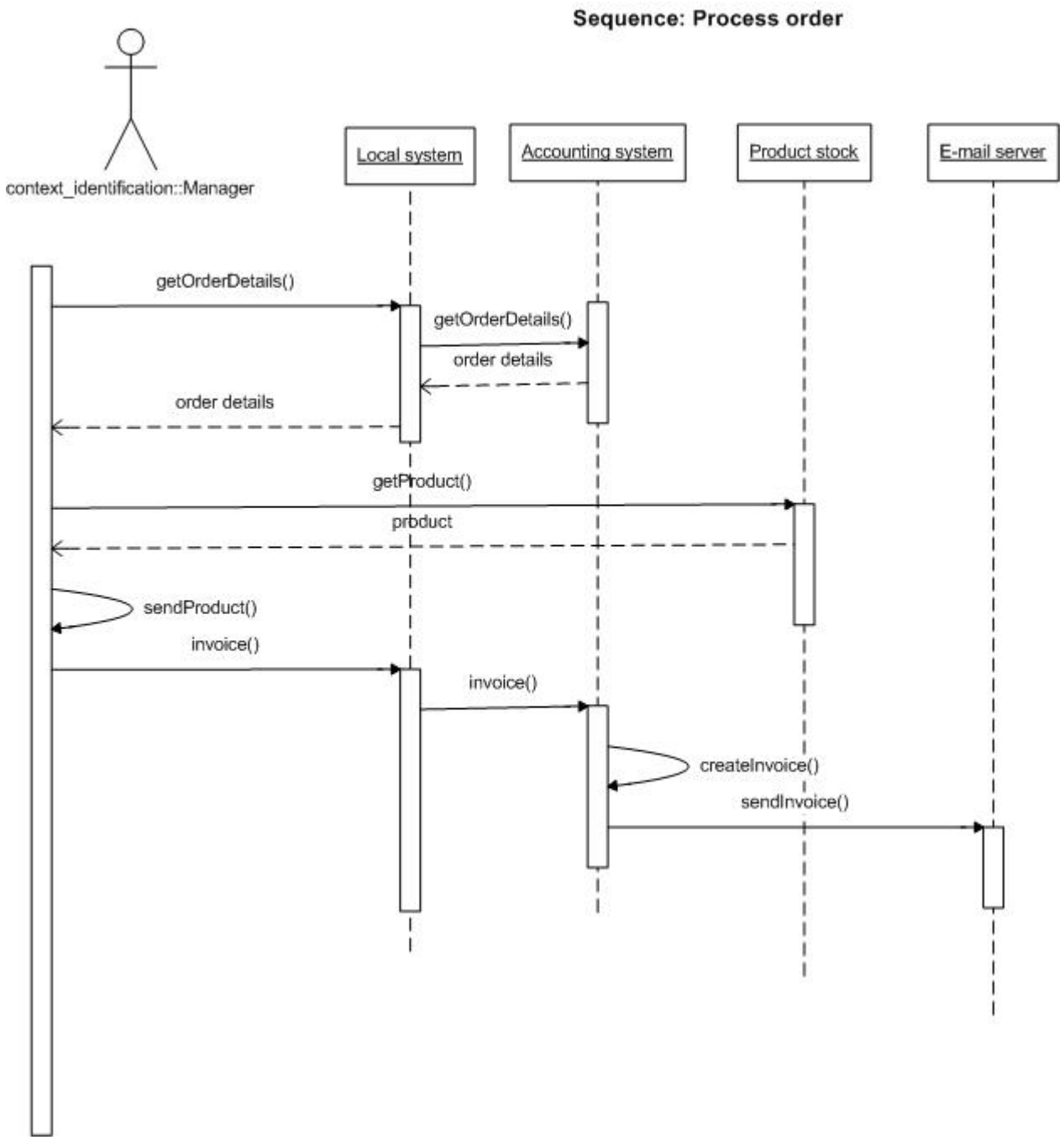

Figure: 26

7.3 Asset identification

The total overview of all identified assets:

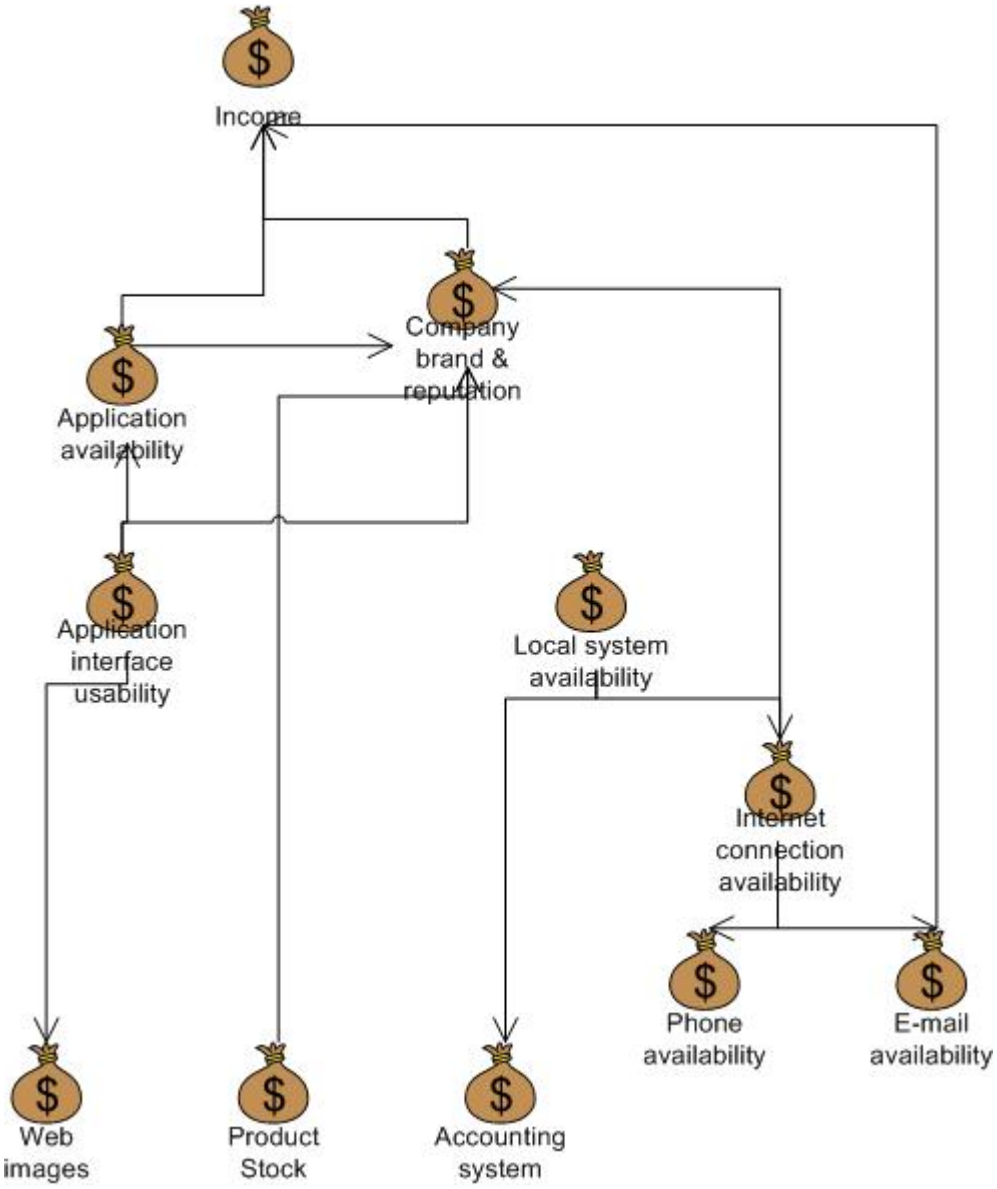

Figure: 27

7.4 System description

Textual specifications of the entities in the system.

Appendix D

The SCORE tool test results

As a part of the evaluation of the SCORE tool, we have conducted a testing of the tool in an experimental setting. We have created a set of specially designed test scenarios explained below that the tool was tested on. Subsequently follows listings of the test records that yield the results on how the tool responded on the test scenarios.

The test scenarios are presented using a frame similar to the one we used to define the SCORE method rules in chapter 4. Its semantics is similar with the first compartment stating the assumptions or the input, except that the second one now displays the result of the calculation.

Test scenarios

We have tested the following scenarios on the SCORE tool:

1. 1-to-1 complete relationships (numerical)
2. Many-to-1 complete relationships (numerical)
3. 1-to-1 complete relationships (linguistic)
4. Many-to-1 complete relationships (linguistic)
5. Interval selection for linguistic representation
6. Inconsistencies (manual)
7. Inconsistencies (automatic)
8. Warnings

Test records

1-to-1 complete relationships (numerical)

We try to perform three simple 1-to-1 incomplete relationships probability calculations.

[1 − to − 1 incomplete relationships]

$$
\begin{array}{ll}
n_1([0,0.1]) & \overset{[0,0.1]}{\rightarrow} \; n \\
m_1([0,0.1]) & \overset{[0.9,1]}{\rightarrow} \; m \\
o_1([0.9,1]) & \overset{[0.9,1]}{\rightarrow} \; o
\end{array}
$$

$$
\begin{array}{ll}
n_1([0,0.1]) & \overset{[0,0.1]}{\rightarrow} \; n(l) \\
m_1([0,0.1]) & \overset{[0.9,1]}{\rightarrow} \; m(k) \\
o_1([0.9,1]) & \overset{[0.9,1]}{\rightarrow} \; o(j)
\end{array}
$$

where

$$
\begin{array}{lcl}
l & = & [0.0, 0.010000000000000002] \\
k & = & [0.0, 0.1] \\
j & = & [0.81, 1.0]
\end{array}
$$

All three calculated probability sets seems reasonable. None of them gets increased probability and the new likelihood sets are all within the boundaries of a legal sample space of $[0, \ldots, 1]$ for probability values. There is however a very small round-off error at the first result (l).

Many-to-1 relationships (numerical)

We try to perform a probability calculation with a disproportionately high number of initiating incidents with a distributed sample space to the extreme.

[Many − to − 1 relationships]

$$
\begin{array}{ll}
n_1([0,0.1]) & \overset{[0.2,0.3]}{\rightarrow} \quad n \\
n_2([0.2,0.3]) & \overset{[0.4,0.6]}{\rightarrow} \quad n \\
n_3([0.4,0.6]) & \overset{[0.7,0.8]}{\rightarrow} \quad n \\
n_4([0.7,0.8]) & \overset{[0.9,1]}{\rightarrow} \quad n \\
n_5([0.9,1]) & \overset{[0-0.1]}{\rightarrow} \quad n
\end{array}
$$

$$
\begin{array}{ll}
n_1([0,0.1]) & \overset{[0.2,0.3]}{\rightarrow} \quad n(l) \\
n_2([0.2,0.3]) & \overset{[0.4,0.6]}{\rightarrow} \quad n(l) \\
n_3([0.4,0.6]) & \overset{[0.7,0.8]}{\rightarrow} \quad n(l) \\
n_4([0.7,0.8]) & \overset{[0.9,1]}{\rightarrow} \quad n(l) \\
n_5([0.9,1]) & \overset{[0-0.1]}{\rightarrow} \quad n(l)
\end{array}
$$

where

$$l = [0.0800000000000002 - 0.21]$$

The examination indicates the rule to be valid, but there are however a slight round-off error.

1-to-1 complete relationships (linguistic)

[1 − to − 1 incomplete relationships]

$$
\begin{array}{ll}
n_1([Veryhigh]) & \overset{[Verylow]}{\rightarrow} \quad n \\
m_1([0,0.1]) & \overset{[0.9,1]}{\rightarrow} \quad m \\
o_1([0.9,1]) & \overset{[0.9,1]}{\rightarrow} \quad o
\end{array}
$$

$$
\begin{array}{ll}
n_1([Verylow]) & \overset{[Verylow]}{\rightarrow} \quad n(l) \\
m_1(Verylow) & \overset{[Veryhigh]}{\rightarrow} \quad m(k) \\
o_1([Veryhigh]) & \overset{[Veryhigh]}{\rightarrow} \quad o(j)
\end{array}
$$

where

$$
\begin{array}{lll}
l & = & [Verylow] \\
k & = & [Verylow] \\
j & = & [Verylow] \\
Verylow & = & [0 - 0.1]
\end{array}
$$

If we compare these results to those provided in the corresponding numerical test above, we see they are identical with respect to their numerical values and how they are mapped to the linguistic.

Many-to-1 relationships (linguistic)

[Many − to − 1 relationships]

$$
\begin{array}{lll}
n_1([Verylow]) & \overset{[Low]}{\rightarrow} & n \\
n_2([Low]) & \overset{[Medium]}{\rightarrow} & n \\
n_3([Medium]) & \overset{[High]}{\rightarrow} & n \\
n_4([High]) & \overset{[Veryhigh]}{\rightarrow} & n \\
n_5([Veryhigh]) & \overset{[Verylow]}{\rightarrow} & n
\end{array}
$$

$$
\begin{array}{lll}
n_1([Verylow]) & \overset{[Low]}{\rightarrow} & n(l) \\
n_2([Low]) & \overset{[Medium]}{\rightarrow} & n(l) \\
n_3([Medium]) & \overset{[High]}{\rightarrow} & n(l) \\
n_4([High]) & \overset{[Veryhigh]}{\rightarrow} & n(l) \\
n_5([Veryhigh]) & \overset{[Verylow]}{\rightarrow} & n(l)
\end{array}
$$

where

$$
\begin{array}{lll}
l & = & [Verylow] \\
Verylow & = & [0 - 0.1]
\end{array}
$$

If we compare these results to those provided in the corresponding numerical test above, we see they are identical with respect to their numerical values and how they are mapped to the linguistic.

Interval selection for linguistic representation

In the top compartment we show which linguistic value we assigned the diagram node. The likelihood definitions are the same as we have used above. The second compartment shows the same nodes, but illustrated with the numerical likelihood value we find in the property view of the current diagram element.

[Interval selection]
$n_1([Verylow])$
$n_2([Low])$
$n_3([Medium])$
$n_4([High])$
$n_5([Veryhigh])$
$n_1([0-0.1])$
$n_2([0.2-0.3])$
$n_3([0.4-0.6])$
$n_4([0.7-0.8])$
$n_5([0.9-1])$

The linguistic maps to their numerical definitions as intended.

Warnings

Not implemented.

List of Symbols and Abbreviations

Abbreviation	Description	Definition
SCORE	simplified CORAS risk estimation	page 3
CORAS	a platform for risk analysis of security critical systems	page 1
AS/NZS	Australian/New Zealand standard	page 8
UML	unified modelling language	page 55
j2se5.0	java™2 platform standard edition 5.0	page 70
JGraph	java graph visualization and layout	page 70
FTA	fault tree analysis	page 9
ETA	event tree analysis	page 13
HazOp	hazard and operability analysis	page 31
FMECA	failure modes, effects and criticality analysis	page 31
EBNF	extended Backus-Naur form	page 8
SRS	software requirements specification	page 49
H1	overall hypothesis 1	page 27
P1	overall success criterion 1	page 29
P2	overall success criterion 2	page 29
P3	overall success criterion 3	page 29
P4	overall success criterion 4	page 30
P5	overall success criterion 5	page 30

List of Figures

1.1 The SCORE project overview 3

2.1 Example of construction of fault tree from BNF notation 11
2.2 Example of construction of event tree from BNF notation 14
2.3 CORAS diagram example 17
2.4 CORAS constructs 18
2.5 Graphical to textual translation of CORAS diagrams 22

4.1 Example model for the SCORE method 40

5.1 The SCORE tool interface 50
5.2 Use case diagram 52

6.1 The property view panel 57
6.2 The likelihood values drop-down box 58
6.3 Likelihood definitions 58
6.4 Choose between numerical and linguistic likelihood expression . . . 59
6.5 Aggregating likelihood value on a 1-to-1 initiate relationship 60
6.6 Aggregating likelihood value on a many-to-1 initiate relationship . . 61
6.7 CORAS overview component diagram 62

6.8 Level 1 composite structure diagram 64
6.9 Level 2 composite structure diagram 64
6.10 Level 3 composite structure diagram 65
6.11 Level 0 interaction diagram - calculate likelihood value 66
6.12 Level 1 sequence diagram - assign likelihood value 67
6.13 Level 2 interaction diagram - calculate likelihood value 68
6.14 Level 3 interaction diagram - calculate likelihood value 69
6.15 Level 4 interaction diagram - calculate likelihood value 70
6.16 the SCORE tool overview class diagram 71

7.1 Target of evaluation and environment 75
7.2 Customer registry confidentiality 79
7.3 E-mail availability 80
7.4 Customer registry confidentiality 81
7.5 E-mail availability 82
7.6 Risk evaluation criteria matrix: Loss of income 83
7.7 Risk evaluation criteria matrix: Inaccessible time 83
7.8 E-mail/customer register confidentiality: Risk estimation numerical . 87
7.9 E-mail availability: Risk estimation numerical 88
7.10 Defining a set 89
7.11 Table of likelihood definitions 89
7.12 Many-to-1 initiate relationship 90
7.13 Inconsistencies 91

A.1 Level 1 composite structure diagram 120
A.2 Level 2 composite structure diagram 121

A.3 Level 3 composite structure diagram 122

A.4 Level 4 composite structure diagram 122

A.5 Level 0 interaction diagram - calculate likelihood value 123

A.6 Level 1 interaction diagram - calculate likelihood value 124

A.7 Level 2 interaction diagram - calculate likelihood value 125

A.8 Level 3 interaction diagram - calculate likelihood value 126

A.9 Level 4 interaction diagram - calculate likelihood value 127

A.10 Level 0 interaction diagram - assign likelihood value 128

A.11 Level 1 interaction diagram - assign likelihood value 129

A.12 Level 2 interaction diagram - assign likelihood value 130

A.13 Level 3 interaction diagram - assign likelihood value 131

A.14 The SCORE tool overview class diagram 132

A.15 The SCORE tool main controller class 132

A.16 The SCORE tool user interface 133

A.17 The SCORE tool likelihood calculator 133

List of Tables

2.1 CORAS constructs definitions 19

3.1 Mappings from fault tree to CORAS diagram 32

3.2 Mappings from event tree to CORAS diagram 33

7.1 Asset table 75

7.2 Frequency values 76

7.3 Probability values 76

7.4 Consequence values:E-mail/customer register confidentiality 76

7.5 Consequence values:E-mail availability 77

7.6 The SCORE tool test results 92